

戴冠平 编著

GoldenGate 企业级运维实战

清华大学出版社

叱咤风云：GoldenGate 企业级运维实战

戴冠平　编著

清 华 大 学 出 版 社
北　京

内 容 简 介

GoldenGate 现在是业内成熟的数据容灾与复制产品，经过多年的发展和完善，现在已经成为业内事实上的标准之一。本书由浅入深地论述了 GoldenGate 的体系和理念，结合作者多年业内专家的从职经验，充分地剖析了 GoldenGate 的核心技术。尤为重要的是，对于 GoldenGate 在实际生产应用中客户系统累积出现的各种典型故障和错误，分门别类地进行了透彻讲解，给出了具体的诊断思路和解决方案，具有非常现实非常重要的指导意义和实战价值。

本书适合作为 GoldenGate 运维技术人员的参考手册，也可以作为高校相关专业师生的学习资料。

图书在版编目（CIP）数据

叱咤风云：GoldenGate 企业级运维实战 / 戴冠平编著. —北京：清华大学出版社，2011.12
ISBN 978-7-302-26761-4

Ⅰ. ①叱…　Ⅱ. ①戴…　Ⅲ. ①互联网络 – 基本知识　Ⅳ. ①TP393.4

中国版本图书馆 CIP 数据核字（2011）第 185705 号

责任编辑：夏兆彦
责任校对：徐俊伟
责任印制：
出版发行：清华大学出版社　　**地　　址：**北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编：**100084
社 总 机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62795954，jsjjc@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：190×260　**印　张：**15　　**字　　数：**375 千字
版　　次：2011 年 12 月第 1 版　　**印　　次：**2011 年 12 月第 1 次印刷
印　　数：
定　　价：　　元

产品编号：043916-01

前　　言

20 世纪 80 年代以来，伴随着网络和互联网的快速发展，中国各行各业的企业级信息系统应用从其体系结构、所采纳的技术和应用本身的复杂性都发生了深刻的变革。企业的 IT 系统日益成为其业务运营的支撑核心。主机、数据库、中间件、存储、网络等基础设施软硬件构成了企业 IT 系统的基石，同时这些不同层面的技术随着企业应用的规模扩张和复杂性增强，对企业的 IT 运维团队提出了越来越高的挑战，这也促使企业用户越来越深刻地认识到运维和服务所起的关键作用，及其带给自己的不菲价值。

在当今信息时代和人才时代，如何能够获得高质量的、快捷便利的 IT 服务是摆在所有企业 IT 运维团队面前的一个迫切需要思考的问题，而单纯求援于目前 IT 大厂商的昂贵服务，成本却常常令大家觉得突兀和吃惊。一个可以作为参考的业内公开数据是，主流企业级软件公司的主要收入，尤其是利润来源，并不是软件销售本身，而是其技术和运维服务；而一般只够支付几年的服务运维费用，就已足够购买一套全新的软件产品。如果有个机会能够以相对公开的方式，针对企业级 IT 的运维进行系统性的知识整理和经验共享，相信这种努力和转移应当是件相当有益处并有意义的工作。

从我个人来讲，毕竟做了这么多年的中间件，很久很久以前，就一直想写这么一本书和大家分享；毕竟曾经写了那么多零零散散的心得，如浩瀚夜空中的点点繁星，错落杂乱；终于有了一大段空闲的时光，能让这些点点滴滴串通起来，努力组成一个有形状有系统的星座。

感谢当年引导和伴随我成长的 BEA 老同事们，让我有机会在中间件的领域里遨游，越游越深，虽然现在应该改称 Oracle 的各位新朋旧友了。也特别感谢联动北方的技术团队，正是你们的敬业精神，才激励我最终坚持把这本书编著出来；也正是你们的辛勤帮助，才有了这本书如此专业详实的内容。

独坐明窗映斜阳，听沧海潮起潮落，观碧空云卷云舒；恰逢母校百年风范依然，学子拳拳赤心依旧，“自强不息，厚德载物”。此书的及时完稿，也算是一丝慰籍和心意了。

作者

2011 年于 Brisbane Manly 海滨

目　　录

第 1 篇　入　门　篇

第 2 篇 基 础 篇

第 3 篇 提 高 篇

第 1 篇

入　门　篇

第 1 章　GoldenGate 概述

GoldenGate 现在是业内成熟的数据容灾与复制产品，经过多年的发展和完善，现在已经成为业内事实上的标准之一。

1.1　GoldenGate 的历史

GoldenGate 公司于 1995 年成立于美国加州旧金山，它的名称源自旧金山闻名于世的金门大桥。两位创始人 Eric Fish 和 Todd Davidson 最初旨在为 Tandem 计算机公司设计一个容错系统，由于 GoldenGate 的健壮性和出色的数据复制功能，银行用它来把 ATM 网络的交易数据发送到 IBM 大型机，后来广泛地应用到金融行业及要求数据复制高效、健壮的各个行业，全球 licences 数量超过 4000。

该公司于 2009 年 9 月被 Oracle 公司收购，在此之前 Oracle 和 GoldenGate 公司就有了长达超过 10 年的合作关系。Oracle 收购 GoldenGate 以后，按照 Oracle 公司的一贯策略，迅速把它和自己的数据库、中间件以及应用集成，依托 Oracle 公司研发技术的优势，并对其做了大量的更新和改进。截止到目前为止，全球已经有五百多家大客户使用 GoldenGate 作为其容灾、复制的解决方案。作为一个企业级的成熟产品，因为其快速、易用、灵活、健壮等特性，越来越多的用户把它作为关键业务系统容灾、复制、同步的首选。随着 Oracle 公司对其不断改进，这个产品会越来越稳定、成熟、可靠，同时也会有更多人去学习它，使用它。

目前 GoldenGate 的最新版本为 11.1，为了和 Oracle 数据库、中间件产品的称谓保持一致，Oracle 称之为 11g，目前 Oracle 公司把它归到 Fusion Middleware，也就是融合中间件产品线中。但实际上它和数据库的联系更加紧密，有经验的 DBA 花较短的时间就可以迅速地掌握它。

1.2　GoldenGate 家族

Oracle GoldenGate 最为常见的家族成员包括 GoldenGate、GoldenGate Director（现更名为 GoldenGate Management Pack，但是绝大部分熟悉 GoldenGate 的人还是习惯性地称其为 GoldenGate Director，为了保证上下文的一致性，这里统一称其为 GoldenGate Director）、GoldenGate Veridata。另外，也有不太常见的 GoldenGate for Mainframe 和 GoldenGate Adapters。

GoldenGate 产品是核心产品，GoldenGate Director 为 GoldenGate 提供友好的 GUI 配置管理界面，而 GoldenGate Veridata 为 GoldenGate 源端和目标端提供数据比对和校验的功能。

注意这三者并不是一个打包的产品，比如如果您购买了 GoldenGate 软件，如果需要图形界面或需要数据校验的功能，就需要额外购买 GoldenGate Director 或者 GoldenGate Veridata。

1.3　GoldenGate 及 Oracle 产品战略

在 Oracle 收购 GoldenGate 以后，与 Oracle 原有的 Data Guard、 ODI 互为补充，共同为企业提供跨平台实时数据同步的解决方案。与此同时，又可以与 Oracle Real Application Cluster、Data Guard 一起为用户提供丰富、灵活的容灾方案及高可用特性，这既是 Oracle 公司当前也是其未来的产品发展策略与方向。

另外有的读者可能注意到了 Oracle 公司对其集成在数据库中的 Streams 的态度的一些变化。Oracle 公司声称对其产品 Streams 将不再做任何重大改进，而是会把 Streams 的一些优秀特性集成到 GoldenGate 产品上来，同时 Oracle 也承诺继续对现有使用 Streams 的客户提供技术支持。与此同时，集成在 Oracle 数据库中的另外一项逻辑复制产品——Data Guard Logical Standby 也遭受了同样的命运。Oracle 对其产品线可整合能力可见一斑。

1.4　GoldenGate 支持的平台以及数据库

截止到笔者撰写本书为止，GoldenGate 几乎支持市面上流行的所有主流操作系统平台和数据库。

在不同的平台生成的 core 文件，有不同的本地堆栈跟踪工具来分析见表 1-1。

表 1-1

数据库产品		操作系统
源端	目标端	及平台
Oracle Database	所有源端支持的数据库	Windows 2000、2003、XP、Server 2008
MySQL	HP Neoview	Linux
IBM DB2	ETL products	Oracle Slaris
Microsoft SQL Server	JMS message queues	HP NoStop
Sybase ASE	Netezza	HP-UX
Ingres	及任何 ODBC 兼容的数据库	HP-TRU64
Timesten		HP-OpenVMS
Teradata		IBM AIX
Enscribe		IBM z/OS
SQL/MP		
SQL/MX		

目前经过 GoldenGate 11.1 认证的主流数据库版本包括以下几种。

- ❑ Oracle 8i (只支持 DML)。

- Oracle 9 i 以上所有的数据库版本（支持 DML 和 DDL)。
- IBM DB2 UDB 8.1、8.2、9.1、9.5 以上版本（只支持 DML）。
- Microsoft SQL Server 2000、2005 和 2008（只支持 DML）。
- MySQL 4.1、5.0（只支持 DML）。
- Sybase ASE 12.5.4、15.0（只支持 DML）。

除 Oracle 数据库 9i 以上版本支持 DDL 以外，8i 及其他数据库均不支持 DDL。

其他经认证的操作系统及数据库版本请参见资料篇的列表。

1.5 GoldenGate 的定位

- 零宕机时间数据库升级和迁移。
- 满足用户亚秒级实时数据的需求。
- 可持续的数据高可用性和实时商务智能。
- 异构平台及跨操作系统实时数据同步。
- 对源系统和目标系统是非侵入式的。

1.6 GoldenGate 技术架构

和传统的逻辑复制一样，Oracle GoldenGate 实现原理是通过抽取源端的 redo log 或者 archive log，然后通过 TCP/IP 投递到目标端，最后解析还原应用到目标端，使目标端实现同源端数据同步。图 1-1 是 Oracle GoldenGate 的技术架构。

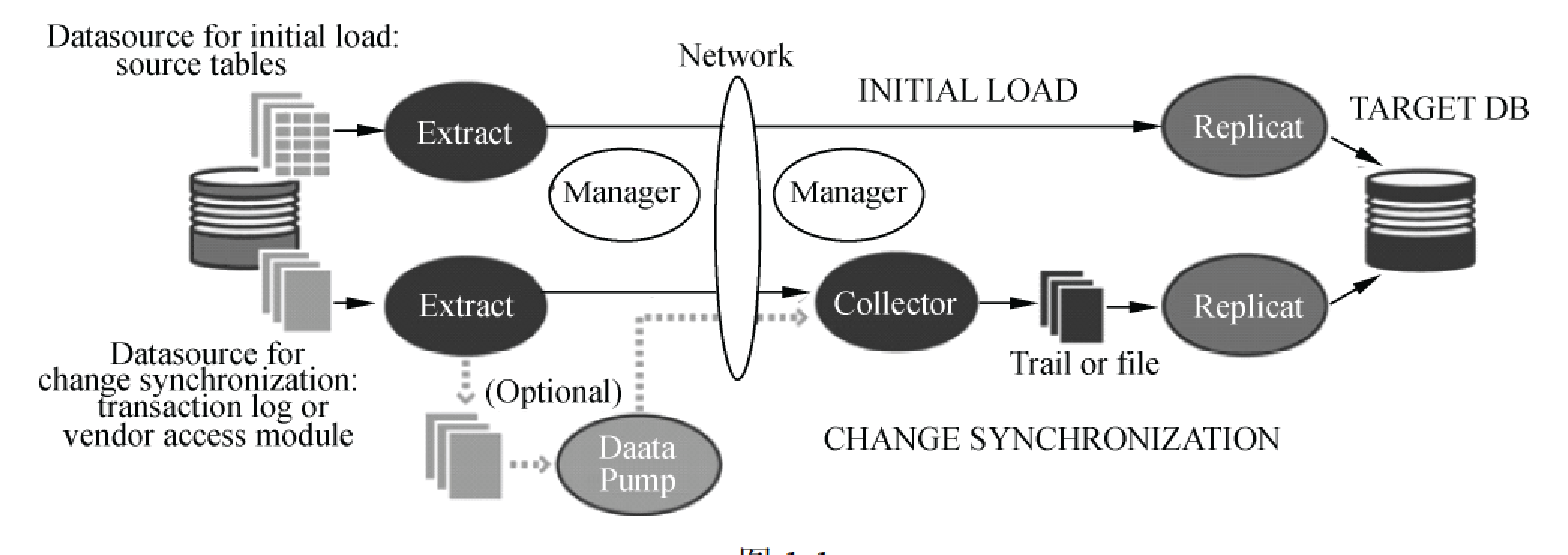


图 1-1

1.6.1 Manager 进程

Manager 进程是 GoldenGate 的控制进程。如果把所有的 Oracle 进程比喻为军队，那么 Manager 就相当于司令。Manager 进程运行在源端和目标端上，它主要有以下几个方面的作用：启动、监控、重启 GoldenGate 的其他进程，报告错误及事件，分配数据存储空间，发布阈值报告等。

每个源端或者目标端有且只能存在一个 Manager 进程。其运行状态有两种即 RUNNING（正在运行）和 STOPPED（已经停止）。

在 Windows 系统上，Manager 进程是作为一个服务来启动的，而在类 UNIX 系统中，Manager 则是一个操作系统进程。

1.6.2　Extract 进程

Extract 运行在数据库源端，负责从源端数据表或者日志中捕获数据。在早期的 GoldenGate 版本中，它通常被称为 Collect 进程。按照其所处的阶段不同，Extract 的作用可以按照时间来划分。

初始数据装载阶段：在初始数据装载阶段，Extract 进程直接从源端的数据表中抽取数据。

同步变化捕获阶段：初始数据同步完成以后，Extract 进程负责捕获源端数据的变化（DML 和 DDL）。

Extract 进程利用其内在的 checkpoint 机制，周期性地检查并记录其读写的位置，通常是写入到一个本地的 trail 文件。这种机制是为了保证如果 Extract 进程终止或者操作系统宕机，重新启动 Extract 进程后，GoldenGate 能够恢复到以前的状态，从上一个断点处继续往下运行，而不会有任何数据损失。

Extract 进程的状态包括 STOPPED（正常停止）、STARTING（正在启动）、RUNNING（正在运行）、ABENDED（Abnomal End 的缩写，表示异常结束）。

1.6.3　Pump 进程

Pump 进程运行在数据库源端，其作用非常简单。如果源端使用了本地的 trail 文件，那么 Pump 进程就会把 trail 以数据块的形式通过 TCP/IP 协议发送到目标端，这通常也是推荐的方式。Pump 进程本质是 Extract 进程的一种特殊形式，如果不使用 trail 文件，那么就是 Extract 进程在抽取完数据以后，直接投递到目标端。

与 Pump 进程相对应的叫做 Server Collector 进程，这个进程不需要引起人们的关注，因为在实际操作过程中无需对其进行任何配置，所以对人们来说它是透明的。它运行在目标端，其任务就是把 Extract/Pump 投递过来的数据块重新组装成 trail 文件，人们称之为远程 trail 文件。

1.6.4　Trail 文件

为了更有效、更安全地把数据库事务信息从源端投递到目标端，GoldenGate 引进 trail 文件的概念。前面提到 Extract 抽取完数据以后 GoldenGate 会将抽取的事务信息转化为一种 GoldenGate 专有格式的文件，然后 Pump 负责把源端的 trail 文件投递到目标端，所以源、目标两端都会存在这种文件，源端存放的 trail 文件叫本地 trail 文件，目标端存放的 trail 文件叫远程 trail 文件。trail 文件存在的目的旨在防止单点故障，将事务信息持久化，并且使用 checkpoint 机制来记录其读写位置，如果故障发生，则数据可以根据 checkpoint 记录的位置来重传。

值得一提的是，trail 文件并不总是必须的。人们可以在配置 Extract 进程的时候通过

TCP/IP 协议直接把日志的信息投递到目标端。但通常并不推荐这么做，因为一旦发生系统宕机或者网络故障，则有可能造成数据的丢失。

1.6.5 Replicat 进程

Replicat 进程，通常也把它叫做应用进程。运行在目标端，是数据传递的最后一站，负责读取目标端 trail 文件中的内容，并将其解析为 DML 或 DDL 语句，然后应用到目标数据库中。

和 Extract 进程一样，Replicat 也有其内部的 checkpoint 机制，保证进程重新启动后可以从上次记录的位置开始恢复，而无数据损失的风险。

它的运行状态和 Extract 进程一致，包括 STOPPED、STARTING、RUNNING、ABENDED。

1.6.6 GGSCI

GGSCI 是 GoldenGate Software Command Interface 的缩写，它提供了十分丰富的命令来对 GoldenGate 进行各种操作，如创建、修改、监控 GoldenGate 进程等。

绝大部分的操作都是通过它来完成的。当然如果您需要 GUI 方式的图形界面来管理，则需要购买 GoldenGate Director。

1.7 GoldenGate 的复制模式

GoldenGate 的模式包括图 1-2 中描述的几种，其中“一对一”是 GoldenGate 最简单的一种模式，也是最常用的模式。

这种模式的一种典型应用就是用于数据容灾，通常源端数据库为生产端，目标端数据库为容灾端。

另外一种应用场景是把源端的 OLTP 系统产生的交易日志传送到目标端，使用 BI 数据仓库或者是 OLAP。

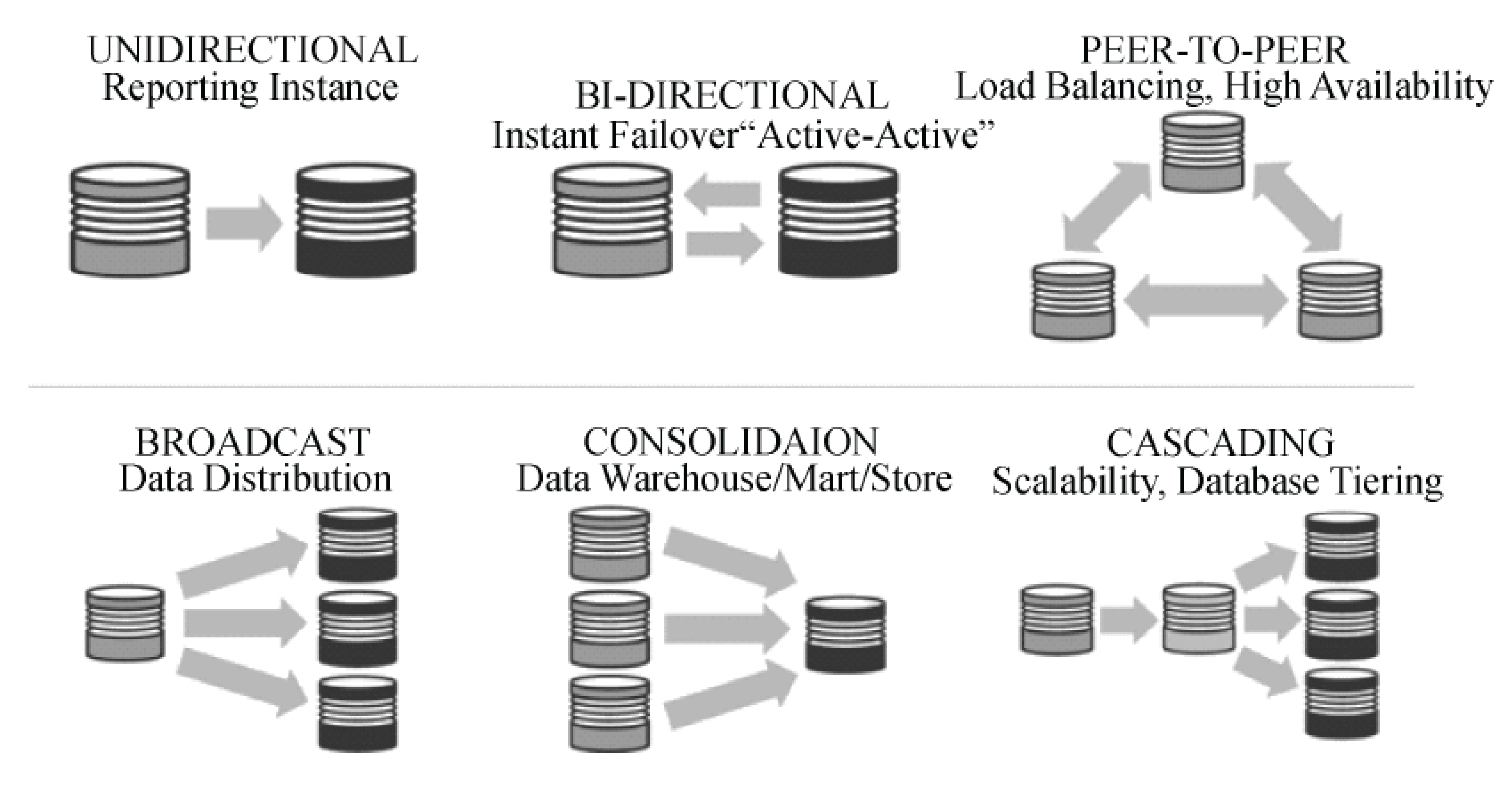


图 1-2

1.8 各种应用模式展示

1.8.1 高可用性：Active-Active

高可用性：Active-Active 如图 1-3 所示。

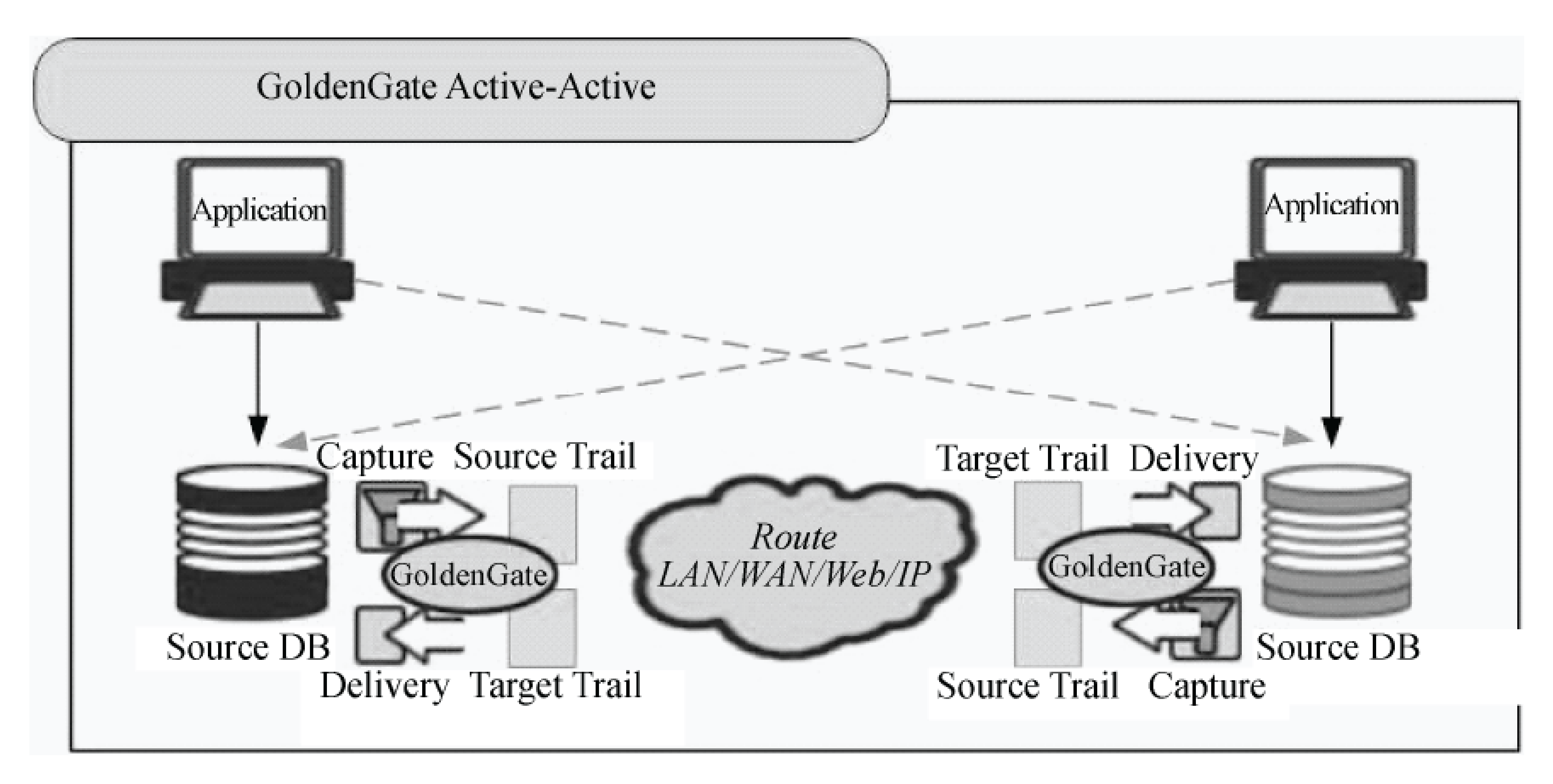


图 1-3

优点：

- 实现连续可用性。
- 实现事务加载和分布式（用内嵌的冲突检测）。
- 提高性能。
- 降低 TCO。

1.8.2 零宕机升降和数据迁移

零宕机升降和数据迁移如图 1-4 所示。

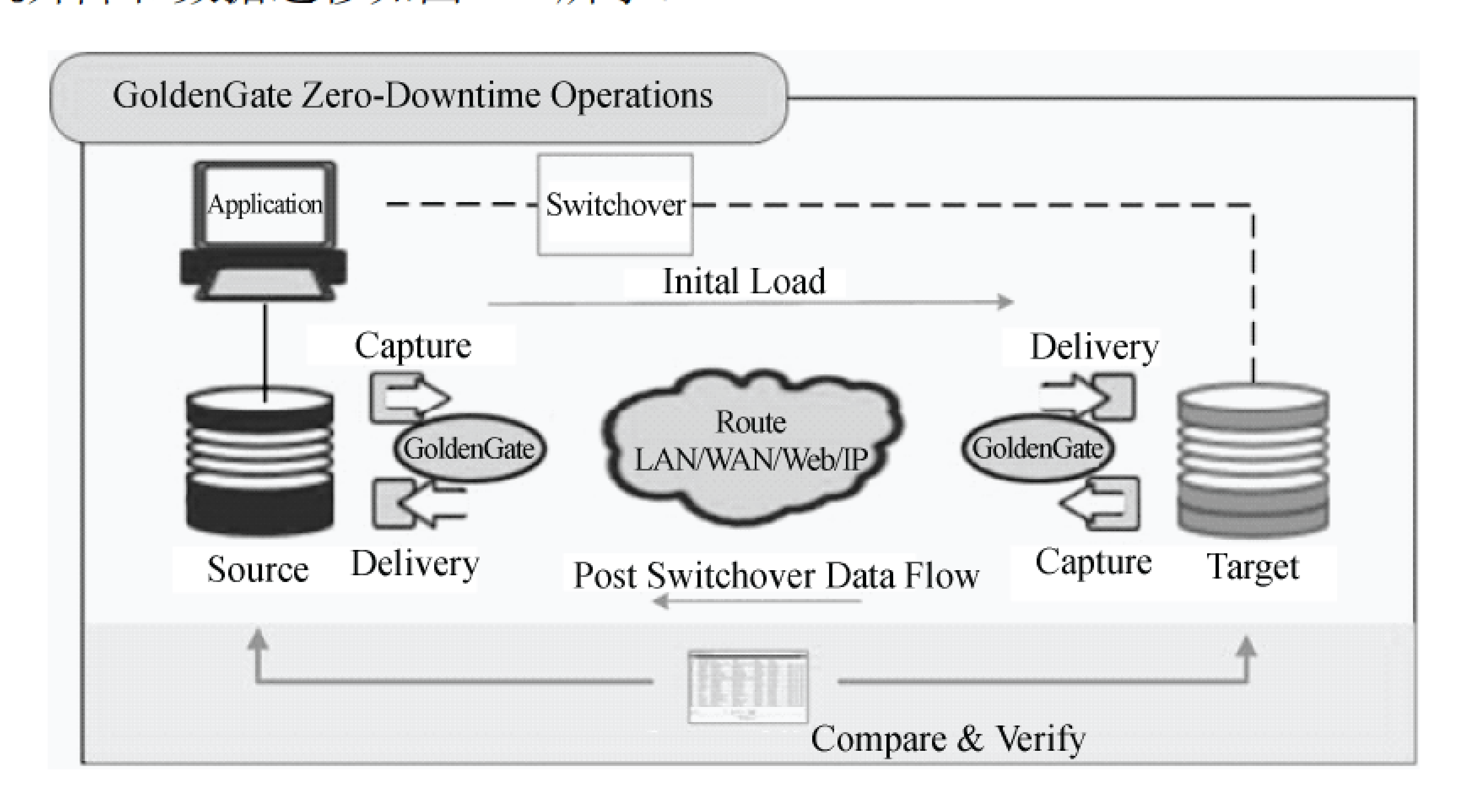


图 1-4

优点：

- ❑ 在硬件，数据库，操作系统或是应用程序升级和数据迁移中消除计划宕机。
- ❑ 减小故障恢复应急风险。
- ❑ 提高用户迁移数据成功率。

1.8.3 数据集成：活动备份

数据集成：活动备份如图 1-5 所示。

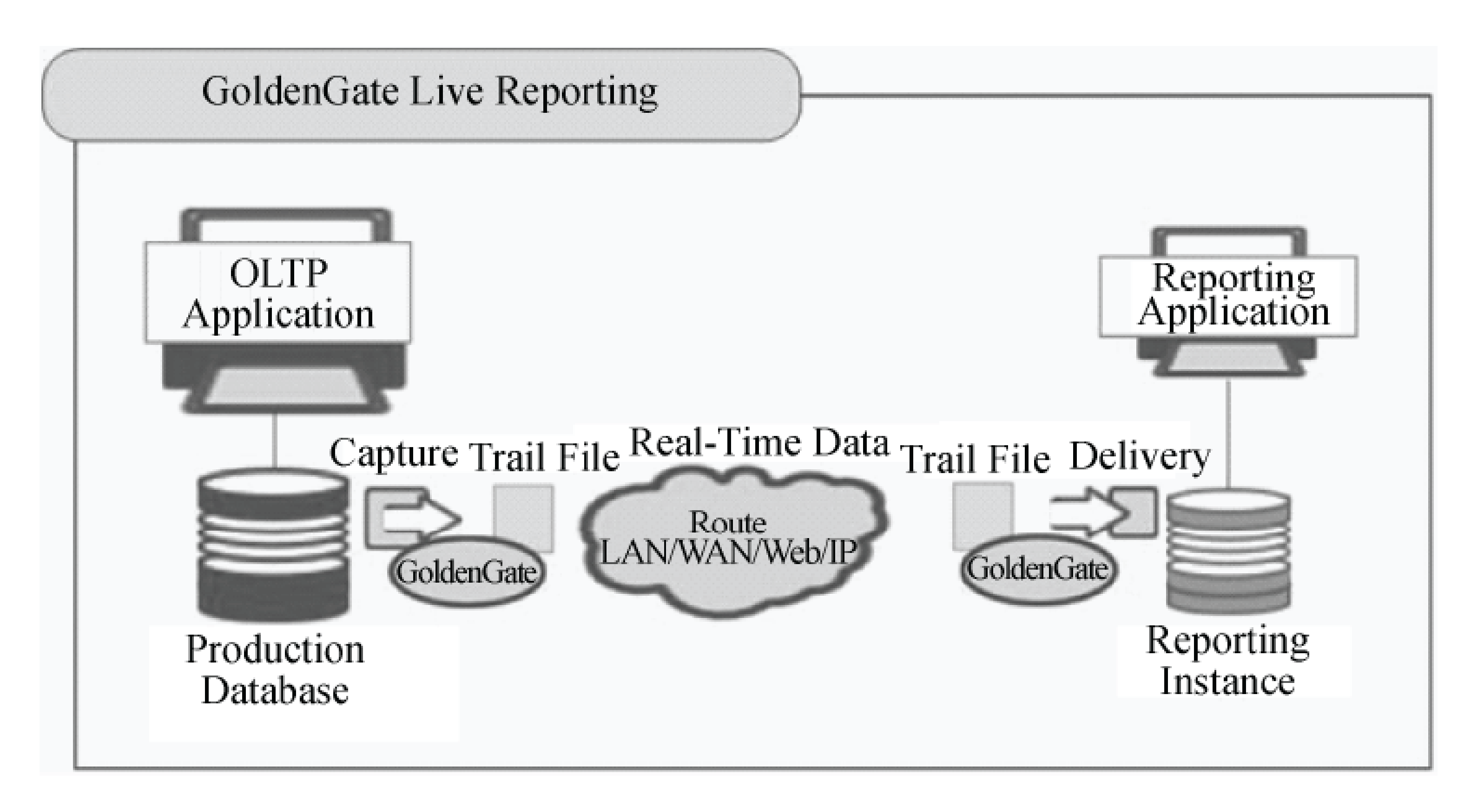


图 1-5

优点：

- ❑ 使用实时数据更好更快的决策。
- ❑ 删除源端报告开销。
- ❑ 减少用户的需求和数据量的成本和规模的增长。
- ❑ 报表需求方面利用有效制度。

1.8.4 商业智能

商业智能如图 1-6 所示。

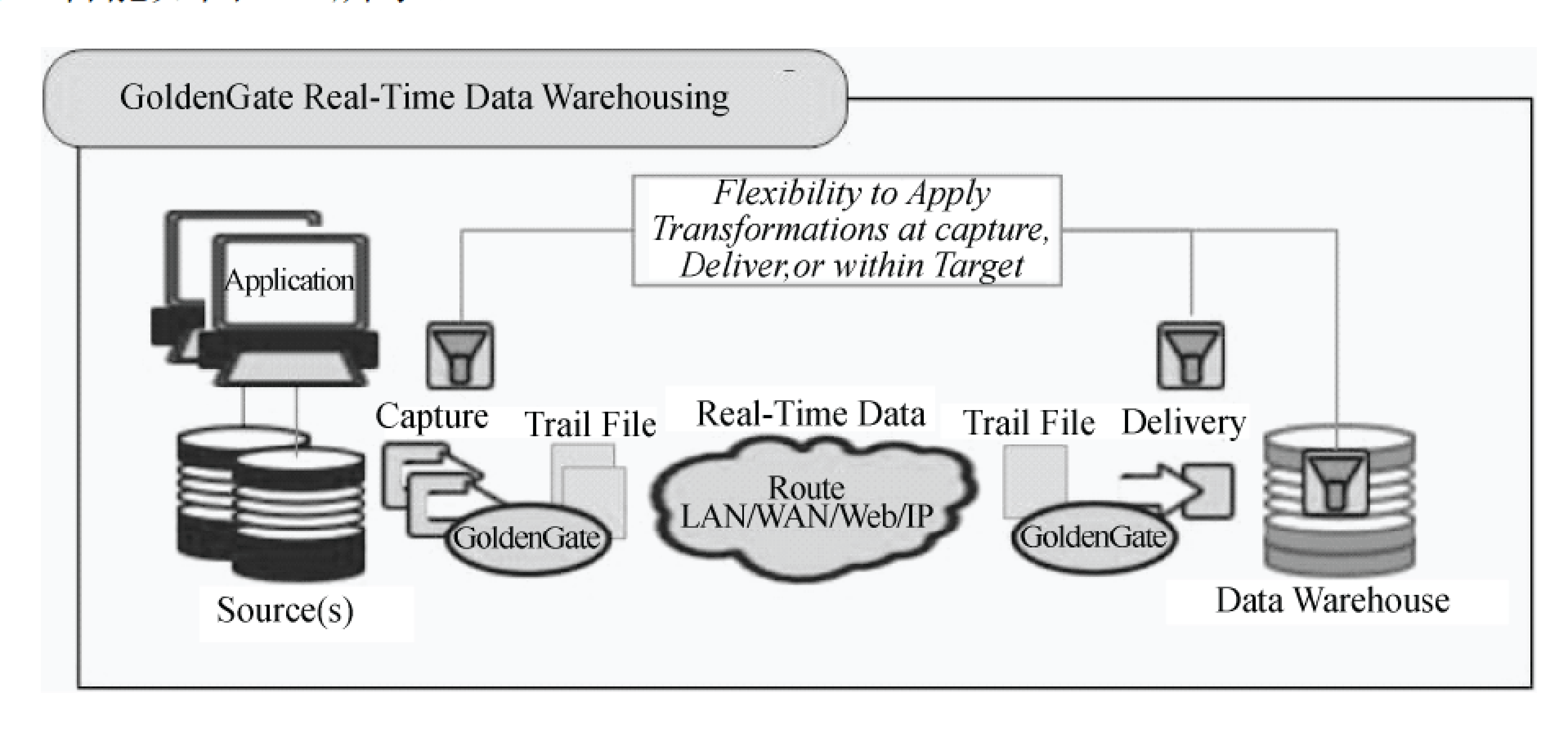


图 1-6

优点：

- 使用实时数据更好更快地决策。
- 消除批量窗口依赖关系。
- 减少源端开销。
- 维护数据质量参照完整性。
- 利用其灵活性转换和集成的 ETL。

1.8.5　事务性数据集成

事务性数据集成如图 1-7 所示。

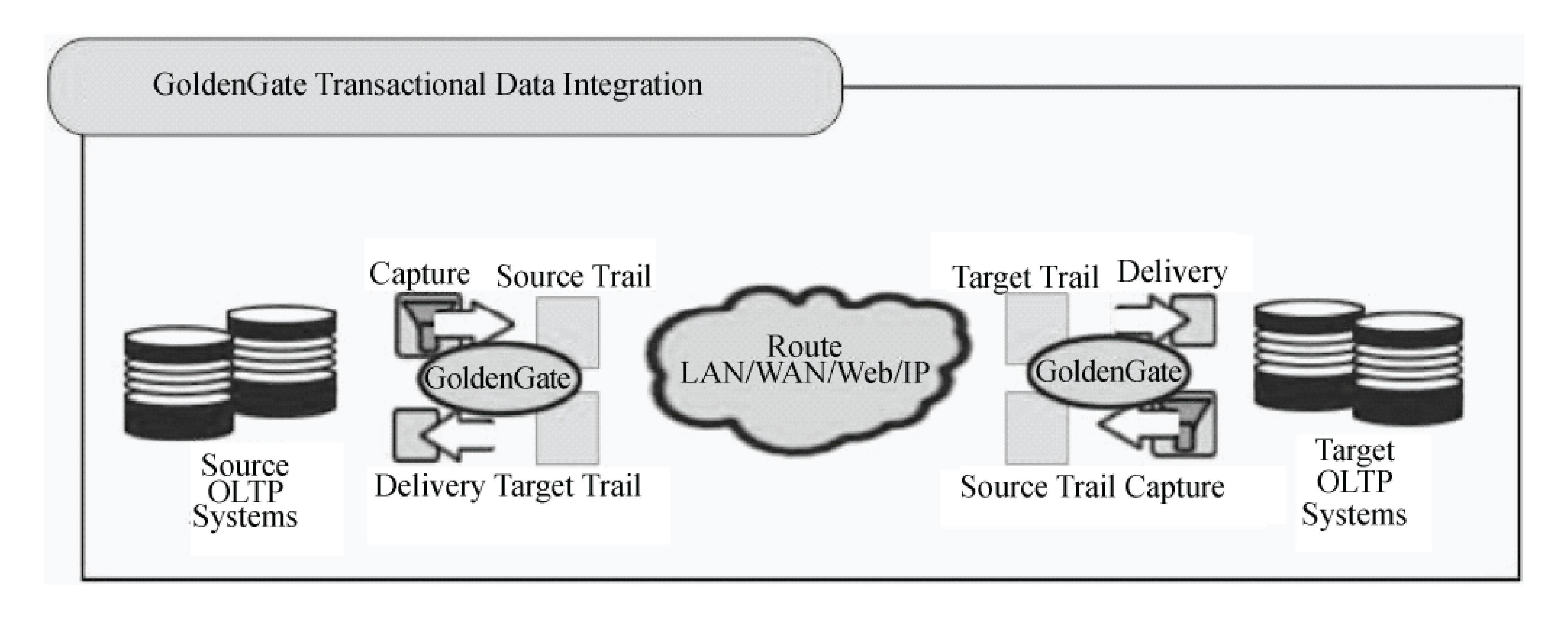


图 1-7

优点：

- 在事务处理系统很容易集成大量实时数据量。
- 减少开销，消除批量窗口。
- 提高可扩展性。
- 增强 SOA 和 EDA 环境（投递基于 JMS 的消息传递系统）。

1.9　GoldenGate 拓扑结构

GoldenGate 有着比较灵活的拓扑结构，如图 1-8 所示。

1.10　GoldenGate 10g/11g 新特性

（1）添加了在 Oracle E-Business Suite、Oracle PeopleSoft and Oracle JD Edwards 等方面的解决方案。

（2）在 Oracle Exadata 数据库上支持更多的数据类型和导入模式。

（3）扩展了对异构的支持。

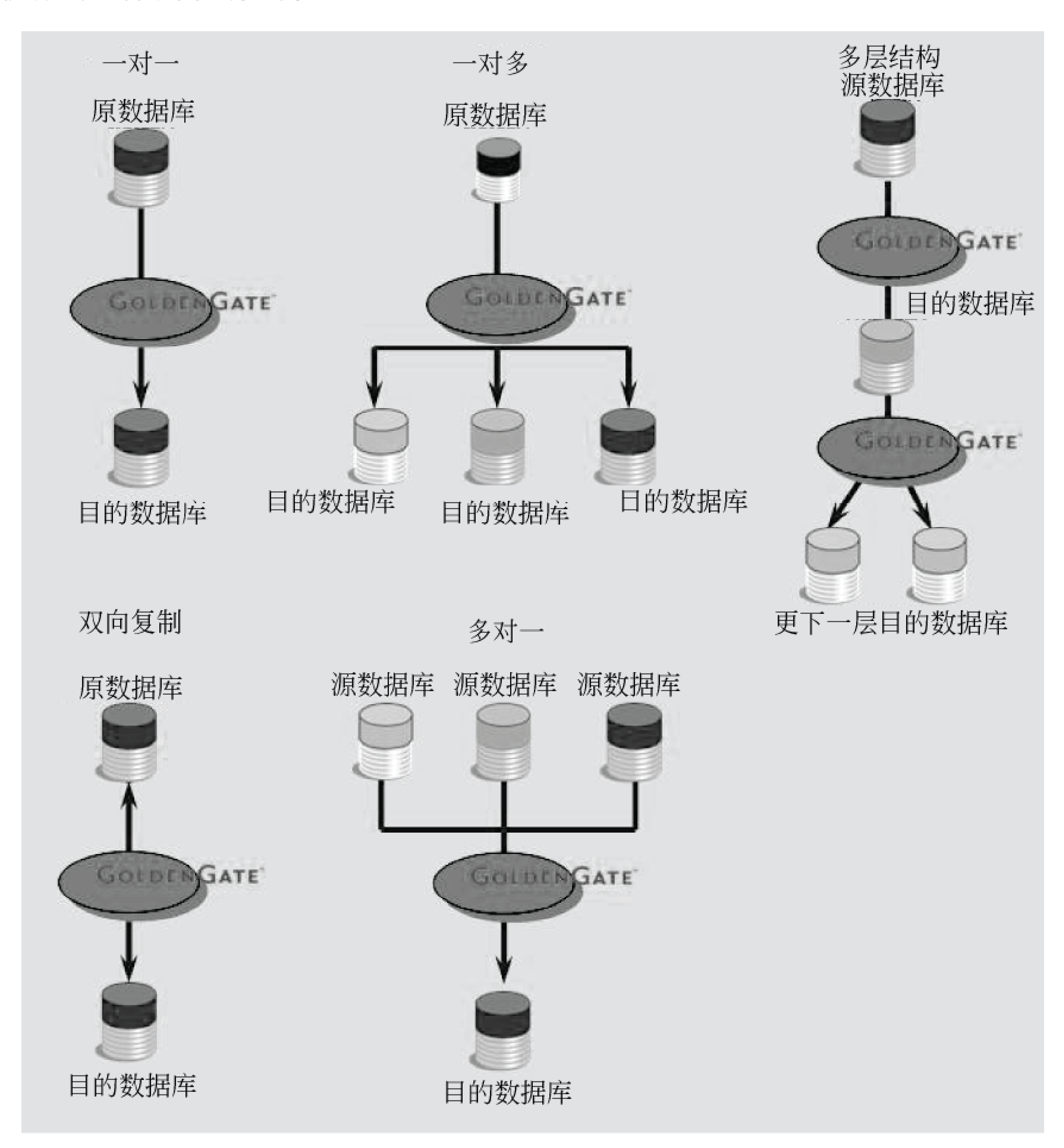


图 1-8

- 基于日志的捕获，投递到 IBM DB2 v9.7。
- 具有投递到 TimesTen databases 的天赋。
- 以 JMS 为基础的消息系统中捕获。
- 自动投递到 IBM 的 DB2。

（4）降低了长事务交易中断所影响的恢复时间。

（5）增强了追踪事务的能力，更容易去排除性能瓶颈。

1.11 GoldenGate 的优势

目前在数据同步/数据复制市场中，除 GoldenGate 外，不乏优秀的产品，它们大部分都是基于 redo 或归档日志进行记录提取实现与源同步。

以下是 Oracle GoldenGate 白皮书中列出的 GoldenGate 的优势见表 1-2。

表 1-2

移动	管理	集成
Speed -Subsecond　Latency	Transaction Integrity	Heterogeneous Data Sources
Volume -Thousands of TPS	Transaction Capture	Mapping
Log-basedCapture	Guaranteed Delivery	Transformation
Native,LocalApply	Conflict Detection,Resolution	Enrichment
Efficient I/O and Band with Usage	Dynamic Rollback	Decoupled Architecture
Bidirectional	Incremental TDM	Table,Row,Column Filtering
Group Transactions	Initial Data Load	XML,ASCCI,SQL Formats
Bulk Operations	GUI-based Monitoring and Configuration	Queue Interface
Compression	Proactive Alerts	Stored Procedures
One-to-Many,Many-to-One	Encrption	User Exits
Cascade	Real-Time Deferred or Batch	ETL Integration
	Event Markers	Java/JMS Integration

1.12　GoldenGate 应用情况

1. 高可用性与容灾

- 容灾与应急备份。
- 消除计划内停机。
- 双业务中心。
- OLTP 和 OLAP 分开。

2. 主数据 数据库移植、升级

3. 实时数据集成

- 数据仓库实时供给。
- 实时报表。
- 政府、企业垂直部门级数据同步。
- 主数据。

第 2 章　Windows 平台 Oracle–Oracle 的单向复制

本章节主要内容为在 Windows 平台下，用 Oracle glodengate 搭建一个单向的 Oracle-Oracle 的复制。在熟悉企业级的复制之前，我们先进行热身，做一个 demo，顺便理清一些基本的概念，对 GoldenGate 好有个全局的概念。

另外用于企业生产环境的 Windows 2003 和 2008 平台也与 Windows XP 大同小异，如果您的平台是 Windows 也可以参考本章的内容。UNIDIRECTIONAL Reporting Instance 如图 2-1 所示。

本章适合于入门及中等水平的读者阅读。

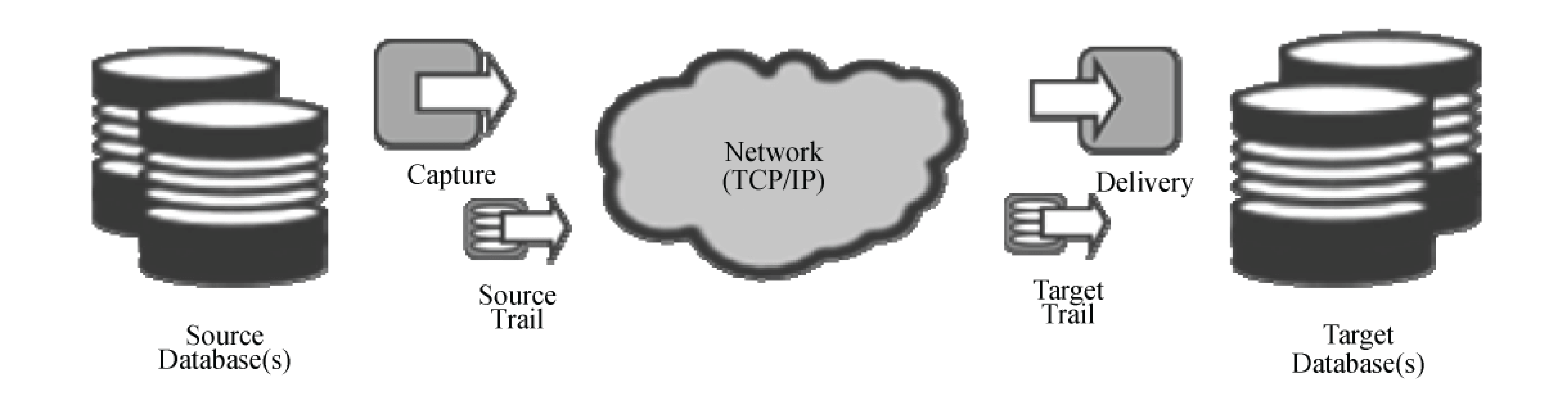


图 2-1

本章节所有试验软件环境见表 2-1。

表 2-1

试验环境	软件版本
操作系统	Microsoft Windows XP Professinal 32bit
数据库	Database version: Oracle Database 10g Enterprise Edition Release 10.1.0.2.0
GoldenGate	Oracle GoldenGate 10.4.0.19 Build 002

2.1　目标概述

GoldenGate 最基本的配置为从源端单向复制到目标端，其中操作系统以及数据库的平台完全一致。这样的场景一般适用于保持目标库数据实时更新，且目标数据库用来检索，如报表或者分析。

2.2　GoldenGate 在 Windows 平台的安装

GoldenGate 实际上使用 C 语言编写，属于跨平台的软件，在不同的平台上差异性很小。如果您购买了 GoldenGate 软件，Oracle 的咨询顾问会提供对应平台和数据库的安装介质。

当然您也可以去 Oracle E-Delivery http://edelivery.oracle.com/下载最新的安装介质。这个网站首次登陆的时候需要注册，注册完以后第二天就可以使用了。进入 E-Delivery 以后，在 Select a Product Pack 中选择 Oracle Fusion Middleware，然后选择相应的平台就可以下载到安装介质及其对应文档了，如图 2-2 所示。

Oracle GoldenGate on Oracle Media Pack v2 for Microsoft Windows (32-bit)

Readme　View Digest

Select	Name	Part Number	Size (Bytes)
Download	Oracle GoldenGate V10.4.0.x for Oracle 10g on Windows 2000, XP, and 2003	V18162-01	8.4M
Download	Oracle GoldenGate V10.4.0.x for Oracle 11g on Windows 2000, XP, and 2003	V18164-01	8.4M
Download	Oracle GoldenGate Documentation on Oracle	V20521-01	7.0M

图 2-2

把安装介质及帮助文档下载后，用解压缩软件解开。本实验是在 C 盘根目录下创建一个叫 gg 的文件夹，然后将软件解压缩到这个文件夹，如图 2-3 所示。

帮助文档解压缩后找到一个名字为 gg_ora_inst_v104.pdf 的 pdf 文档，这个 pdf 会有非常详细的安装帮助与说明。

安装之前，你需要安装好 Oracle 数据库，并且需要安装 Microsoft Visual C++ 2005 SP1 Redistributable，这是 Windows 环境下 Visual C++库的运行组件。GoldenGate 运行的时候会用到它的一些库，所以不安装的话，可能导致“系统无法执行指定的程序”的错误。

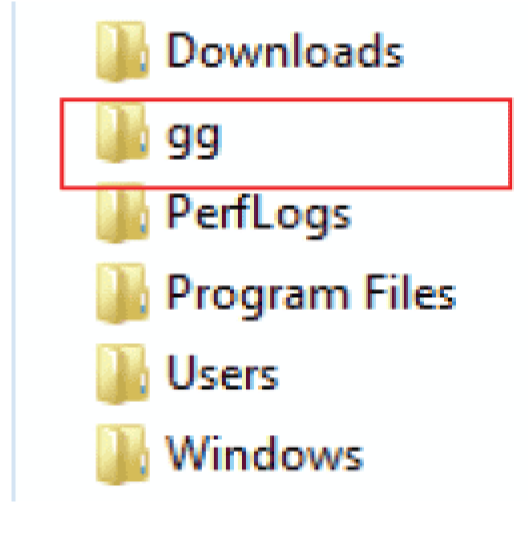


图 2-3

注 意

事实上 GoldenGate 的安装 tar 包在类 UNIX 平台命名的规则十分清晰，从 tar 包的名字就可以得出操作系统、数据库和 GoldenGate 的一些基本信息。

例如 tar 包的名称为 ggs_Linux_x64_sybase15_64bit_v11_1_1_0_1_004.tar，就包含了以下信息。

- 操作系统： Linux 64bit。
- 数据库：Sybase 15.0 64bit。
- GoldenGate：11.1.1.0.1 build 004。

2.2.1 安装 GoldenGate 软件

1. 设置环境变量

安装之前，需要先设置 Oracle 数据库的环境变量。

右击我的电脑，在弹出的快捷菜单中选择“属性”命令，在打开的对话框的“高级”选项卡中单击“环境变量”按钮，然后新建一个系统变量 ORACLE_HOME，并把其对应的 ORACLE_HOME 对应的路径值输入，再单击“确定”按钮，然后运行 cmd，在命令行输入 echo %ORACLE_HOME% ，如果返回的结果正确，那么 ORACLE_HOME 环境变量的设置就完成了，再按照同样的步骤，设置环境变量 ORACLE_SID，如图 2-4 所示。

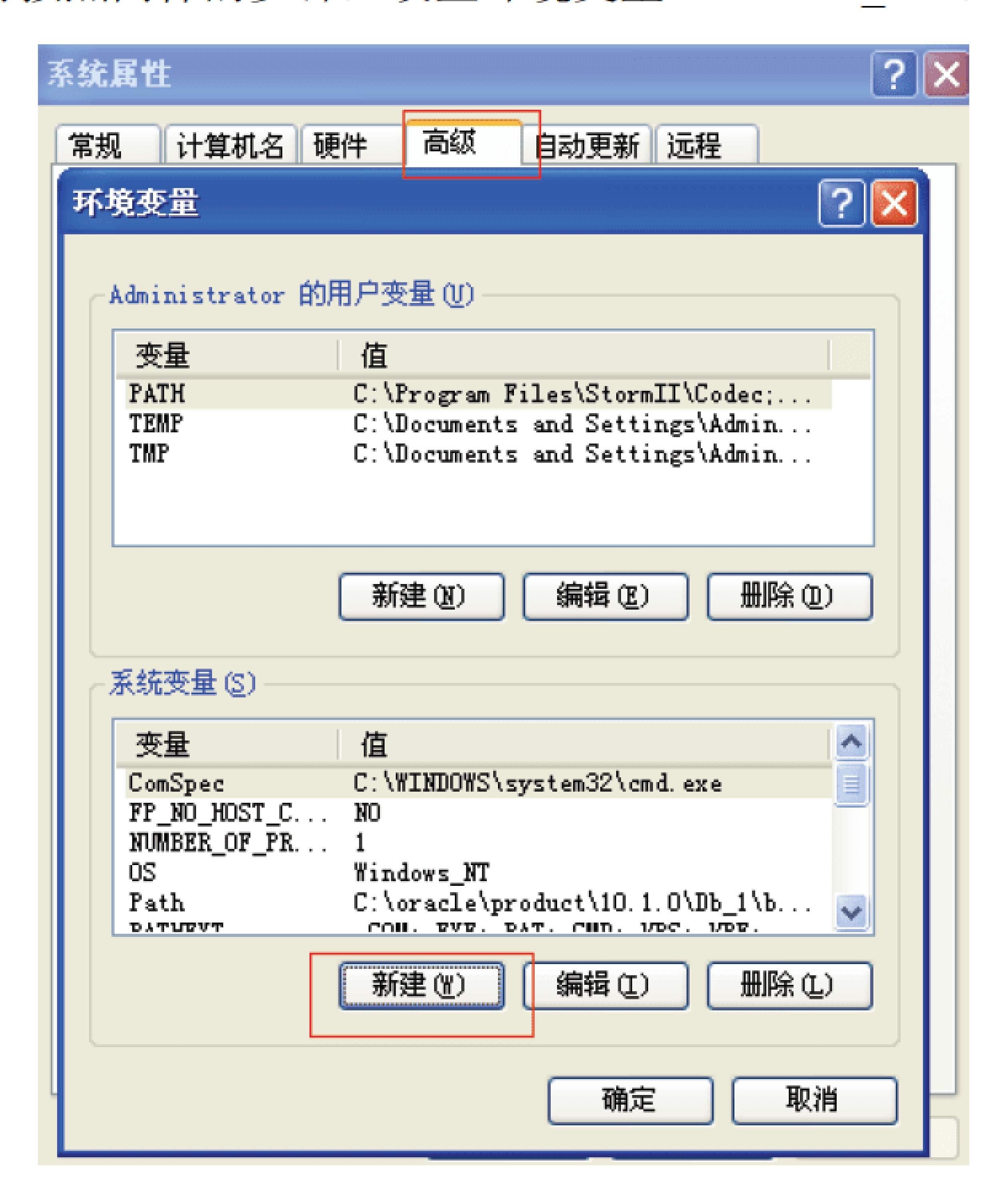


图 2-4

2. 安装 GoldenGate 文件

把下载好的安装介质 V18162-01.zip 的内容解压缩到事先建好的目录：C:\gg 下，然后在 cmd 里，cd 到 gg 目录下，运行 GGSCI 命令，创建 GoldenGate 工作目录：

示例 2-1：

```
C:\gg>GGSCI
GGSCI (source) 1> create subdirs
```

然后使用 exit 退出 GGSCI 命令行。这样 GoldenGate 软件安装就完成了，如图 2-5 所示。

```
GGSCI (source) 1> create subdirs

Creating subdirectories under current directory C:\gg

Parameter files                C:\gg\dirprm: created
Report files                   C:\gg\dirrpt: created
Checkpoint files               C:\gg\dirchk: created
Process status files           C:\gg\dirpcs: created
SQL script files               C:\gg\dirsql: created
Database definitions files     C:\gg\dirdef: created
Extract data files             C:\gg\dirdat: created
Temporary files                C:\gg\dirtmp: created
Veridata files                 C:\gg\dirver: created
Veridata Lock files            C:\gg\dirver\lock: created
Veridata Out-Of-Sync files     C:\gg\dirver\oos: created
Veridata Out-Of-Sync XML files C:\gg\dirver\oosxml: created
Veridata Parameter files       C:\gg\dirver\params: created
Veridata Report files          C:\gg\dirver\report: created
Veridata Status files          C:\gg\dirver\status: created
Veridata Trace files           C:\gg\dirver\trace: created
Stdout files                   C:\gg\dirout: created


GGSCI (source) 2>
```

图 2-5

下面来介绍其中一些比较重要的目录。

（1）dirchk：用来存放检查点（Checkpoint）文件。

（2）dirdat：用来存放 Trail 文件，以后详述。

（3）dirdef：用来存放通过 DEFGEN 工具生成的源或目标端数据定义文件。

（4）dirpcs：用来存放进程状态文件。

（5）dirprm：用来存放配置参数文件。

（6）dirrpt：用来存放进程报告文件。

（7）dirsql：用来存放 SQL 脚本文件。

（8）dirtmp：当事务所需要的内存超过已分配内存时，默认存储在这个目录。

更详细的内容请参考 Oracle® GoldenGate Oracle Installation and Setup Guide。

3．把 manager 进程添加到 Windows 服务

在默认情况下，Manager 进程并没有作为服务安装并且可以运行在本地或者域账户下。如果使用这种方式，当用户注销后，Manager 进程就会强制停止。

如果把 Manager 进程作为 Windows 服务安装，那么 Manager 就会与用户的连接无关。你可以配置它随着操作系统启动或者手工启动。在非 Windows 集群下这是可选的，但是强烈建议把 Manager 进程安装为 Windows 服务。

在 Windows 集群环境下，则必须把 MGR 进程添加到 Windows 服务，因为一台服务器出现故障时，只有通过服务才能将其自动切换到备用服务器上。

服务名可以手动指定，也可以为默认值（默认值为 GGSMGR）。用如下方法手动指定服务名：在命令行模式下切换到 gg 目录，用 GGSCI 命令进入 GGSCI 交互界面，然后输入 EDIT PARAMS ./GLOBALS，（注意：/GLOBALS 最好使用大写）在弹出的记事本编辑器中，输入以下代码：

示例 2-2：

```
MGRSERVNAME  GGMGR
```

保存并退出后，回到 Windows 命令提示符，在 gg 根目录用 install 命令添加服务：install addservice ，然后手动命名的服务，这样就添加成功了。当然也可以直接使用默认的服务名 GGSMGR——直接在 gg 目录输入 install addservice，如图 2-6 所示。

```
C:\gg>install addservice

Service 'GGSMGR' created.


Install program terminated normally.


C:\gg>
```

图 2-6

在 cmd 里也可以直接输入 install help 查看该命令的帮助。

4．GoldenGate 实用程序

运行 GGSCI（GoldenGate Software Command Interface）程序，就能进入到 GoldenGate 的交互界面了。里面提供了非常丰富的配置和管理 GoldenGate 所使用到命令可以在 GGSCI 提示符下输入 help。

下面介绍一下帮助的使用。运行 GGSCI 后，输入 help，就能看到 GoldenGate 的绝大多数命令了。其中左边的是 object，右边的是 command。所有的命令无需死记硬背，非常方便。在查询一个命令的具体用法的时候，可以使用 help <command> <object> 得到一个命令详细的语法，例如示例 2-3，操作如图 2-7 所示。

```
GGSCI (source) 3> help

GGSCI Command      Summary
SUBDIRS            CREATE SUBDIRS
ER                 INFO ER, KILL ER, LAG ER, SEND ER, STATUS ER,
                   START ER, STATS ER, STOP ER
EXTRACT            ADD, ALTER, CLEANUP, DELETE, INFO, KILL,
                   LAG, SEND, START, STATS, STATUS, STOP
EXTTRAIL           ADD, ALTER, DELETE, INFO
GGSEVT             VIEW
MANAGER            INFO, REFRESH, SEND, START, STOP, STATUS
MARKER             INFO
PARAMS             EDIT, VIEW
REPLICAT           ADD, ALTER, CLEANUP, DELETE, INFO, KILL,
                   LAG, SEND, START, STATS, STATUS, STOP
REPORT             VIEW
RMTTRAIL           ADD, ALTER, DELETE, INFO
TRACETABLE         ADD, DELETE, INFO
TRANDATA           ADD, DELETE, INFO
Database           DBLOGIN, LIST TABLES,
                   ENCRYPT PASSWORD
DDL                DUMPDDL
CHECKPOINTTABLE    ADD CHECKPOINTTABLE, DELETE CHECKPOINTTABLE,
                   CLEANUP CHECKPOINTTABLE, INFO CHECKPOINTTABLE
Miscellaneous      FC, HELP, HISTORY, INFO ALL, INFO MARKER, OBEY,
                   SET, SHELL, SHOW, VERSIONS, !


For help on a specific command, type HELP <command> <object>.

Example: HELP ADD REPLICAT

GGSCI (source) 4>
```

图 2-7

示例 2-3：

```
help  add  replicat
```

大致分类请参看表 2-2。

表 2-2

CGSCI 命令	描述
Manager commands	用于启动和管理 MGR 进程
Extract commands	创建和管理 Extract 进程组
Replicat commands	创建和管理复制进程组
ER commands	以组的形式统一控制抽取进程组与复制进程组
Trail commands	将 trail 文件与抽取进程相关联
Parameter commands	编辑或者改变参数文件内容
Database commands	运行与数据库相关的命令
Trandata commands	对要传输的表添加额外的日子信息供复制进程使用
Checkpoint table commands	创建与管理 GG 的检查点表
Oracle trace table commands	创建和管理 trace 表以阻止传输数据形成环路
DDL commands	与 DDL 同步相关的命令
Miscellaneous commands	杂项命令，如 shell、create subdirs 等

2.2.2　配置 Oracle 数据库

在配置 GoldenGate 之前，还需要对数据库进行一些特殊的设置，下面一一说明。

1．调整归档模式

GoldenGate 的原理是基于对日志变化的捕获（CDC），所以 Oracle 的 redo 对于 GoldenGate 至关重要。为了保证 GoldenGate 能读取到完整的事务日志，必须打开归档。

在数据库负载较大的情况下，redo 会频繁切换日志组，大家知道，redo 日志组在 Oracle 中有限且会被重复利用的。如果打开归档，被切换过的 redo log 就会被归档为 archive log，这样即使一个事务过长，等到提交的时候，部分可能已经归档，这时 GoldenGate 就会到 archive log 中查找对应的记录，从而保证了信息的完整性。

Oracle 10g 版本的 Oracle 打开归档比较简单：首先，使用 sysdba 用户登录在 SQLPLUS 下执行以下命令，确认归档是否已经开启：

示例 2-4：

```
SQL>archive log list;
```

查看 Database log mode 字段的值，如果是 No Archive Mode，则表示没有开归档。这时如果要开启归档，则需要先关闭数据库，然后启动到 startup mount 状态，再打开归档：

示例 2-5：

```
SQL>shutdown immediate;
SQL>startup mount;
SQL>alter database archivelog;
```

2．打开数据库级别的补充日志（supplemental log）

对于逻辑复制，怎样把源端操作准确无误地反映到目标端呢？大家知道：逻辑复制源端数据库和目标端数据库的数据块结构可能是完全不一样的，因为无法通过 rowid 的信息进行精确查找与准确定位，这时 supplemental log 就派上用场了。

当数据库启用 supplemental logging 之后，对于修改操作，Oracle 就会同时附加一些能够唯一标识修改记录的列到 redo log 中。如果这个表是有主键或唯一键的表，只需要附加主键或唯一键的信息即可，这样生成的 redo 日志量是最少的。如果某些表可能无法创建主键或者唯一键，或者数据表本来不存在主键/唯一键，这种情况下 Oracle 会将所有列都作为附加信息记录到 redo 中，那么 redo 就可能增长很快，同时对性能产生较大的负面影响。

所以 Oracle 建议所有需要复制的表都存在主键或者唯一键。

supplemental logging 可以在数据库级设置，也可以精确到表级设置，对于数据库级有两种类型：minimal logging 和 identification key logging，其主要区别就在于写入 redolog 中的数据详尽程度不同。

从上述分析可以得出：GoldenGate 要准确地知道源端的数据修改了哪些列，就需要更为详细的日志信息，所以需要数据库开启 supplemental log。

可以按照下列方式打开数据库级别的 supplemental log。

首先确认数据库是否开启了 supplemental log，如果没有，则开启，操作如图 2-8 所示。

```
SQL> select supplemental_log_data_min from v$database;

SUPPLEME
--------
NO

SQL> alter database add supplemental log data;

Database altered.

SQL> select supplemental_log_data_min from v$database;

SUPPLEME
--------
YES

SQL> _
```

图 2-8

请注意，对于操作

示例 2-6：

```
SELECT supplemental_log_data_min FROM v$database;
```

如果结果返回 YES 或 IMPLICIT 则说明已开启最小补全日志，另外如果使用 ALL、PRIMARY、UNIQUE 或 FOREIGN 补全日志时，最小补全日志会默认开启（即检查结果为 IMPLICIT）。

3．创建 GoldenGate 管理用户

在源端和目标端创建 GoldenGate 管理用户，并授予下列权限：

示例 2-7：

```
SQL>create user ggs identified by ggs default tablespace users temporary
tablespace temp;
SQL>grant connect, resource, unlimited tablespace to ggs;
SQL>grant execute on utl_file to ggs;
```

以上只是基本权限，在源、目标两端都要执行。
另外，在源端还需要授予 ggs 用户以下权限：
示例 2-8：

```
SQL>grant connect,resource to ggs;
SQL>grant select any dictionary,select any table to ggs;
SQL>grant alter any table to ggs;
SQL>grant flashback any table to ggs;
SQL>grant execute on DBMS_FLAHBACK TO ggs;
```

在目标端需要授予 ggs 用户以下权限：
示例 2-9：

```
grant insert anytable to ggate;
grant delete any table to ggs;
grant update any table to ggs;
```

上面的 insert、delete、update 没有指定模式，表示在所有模式都能进行 insert、delete、update 操作。

如果对权限要求不那么严格，最简单的办法就是直接授予 GoldenGate 管理用户 dba 权限：

示例 2-10：

```
SQL>grant dba to ggs;
```

4．添加表级 trandata

这里的表级 trandata 就是指表级的 supplemental log。

表级 supplemental log 需要在数据库级别最小 supplemental log 打开的情况下才起作用。如果数据库没有开启 minimal supplemental log，即使指定了表级 supplemental log，实际在 redo log 输出的过程中描述的记录仍只记录 rowid 和相关列值，所以还需要同时开启表级的 trandata。

添加表级的 trandata 可以理解为你需要将源端的哪些 schema 下的哪些表传输到目标库中，那么就需要你手动添加这些 trandata。

按照上面的内容创建完 GoldenGate 管理用户并授予必要的权限之后，就可以使用 dblogin 登录数据库，然后添加特定表的 trandata。

操作步骤如下：

在 scott 用户下创建一张表 demo，用来作为需要同步的表，操作如图 2-9 所示：

```
SQL> create table demo(id number primary key,ename varchar2(10));

Table created.

SQL>
```

图 2-9

对这个表添加 trandata，操作如图 2-9 所示：

```
GGSCI (source) 3> dblogin userid ggs,password ggs
Successfully logged into database.

GGSCI (source) 4> add trandata scott.demo

Logging of supplemental redo log data is already enabled for table SCOTT.DEMO.

GGSCI (source) 5>
```

图 2-10

提 示

在对表添加 trandata 的时候，表名可以使用通配符。例如如果要添加 scott 用户下所有的表，则语句可以这样写：

示例 2-11：

```
GGSCI(source 4)add trandata scott.*;
```

5. 添加 checkpoint 表

在目标端配置复制进程 Replicat 之前，需要在目标端的数据库中创建一个 checkpoint 表。这个 checkpoint 表是基于 GoldenGate checkpoint 文件的，它记录了所有 GoldenGate 可恢复的 checkpoint 以及 sequence。

尽管这个操作也不是必须的，但是 Oracle 强烈建议使用它，因为它可以使得 checkpoint 包含在 Replicat 的事务中，保证了可以从各类失败场景中恢复。

添加的步骤为在目标机器上编辑 GLOBALS 文件，添加一行，checkpointtable ggs.checkpoint，然后使用 dblogin 登录数据库，使用 add checkpoint 命令添加 checkpoint 表。

操作步骤如图 2-11 所示：

```
GGSCI (target) 3> edit params ./GLOBALS


GGSCI (target) 4> dblogin userid ggs,password ggs
Successfully logged into database.

GGSCI (target) 5> add checkpointtable ggs.checkpoint

Successfully created checkpoint table GGS.CHECKPOINT.
```

图 2-11

6. 关闭 recyclebin

在 GoldenGate 10g 中如果要使用 DDL 复制，则必须先关闭回收站。本实验不需要 DDL

支持，不需要配置。另外 recyclebin 在 10.1 和 10.2 中关闭的方式稍微有所不同。

Recyclebin 在 10.1 中是隐含参数，本实验中用到的数据库是 10.1 版本，所以关掉 recyclebin 的命令为：

示例 2-12：

```
SQL>ALTER SYSTEM SET "_recyclebin" = false;
```

如果数据库的版本为 10.2，则只执行的命令是：

示例 2-13：

```
SQL>ALTER SYSTEM SET recyclebin = OFF;
```

另外在最新的 GoldenGate 11g 中即使使用其 DDL 复制的功能也不需要关闭 recyclebin 了。

2.2.3　GLOBALS 参数文件

在这之前已经配过了两组 GLOBALS 参数，一个是添加 Windows 服务的，另外一个是用来添加 checkpoint 表，所以应该不再陌生了。

GLOBALS 文件中存的参数对全局起作用（注意这里 GLOBAL 总是大写），参数可以有 mgrservname、checkpointtable、ggschema、ddltable、markertable、outputfileumask。其中后几个参数是与 DDL 相关的参数，后面在 DDL 复制的场景会有更详细的介绍，也可以参考官方手册 Oracle GoldenGate Reference Guide。

2.3　配置 GoldenGate 进程组

一般常用的进程包括在源端配置 MGR 进程、Extract（抽取）进程、Pump 进程、在目标端配置 MGR 管理进程、Replicat（复制）进程。各进程总览如图 2-12 所示：

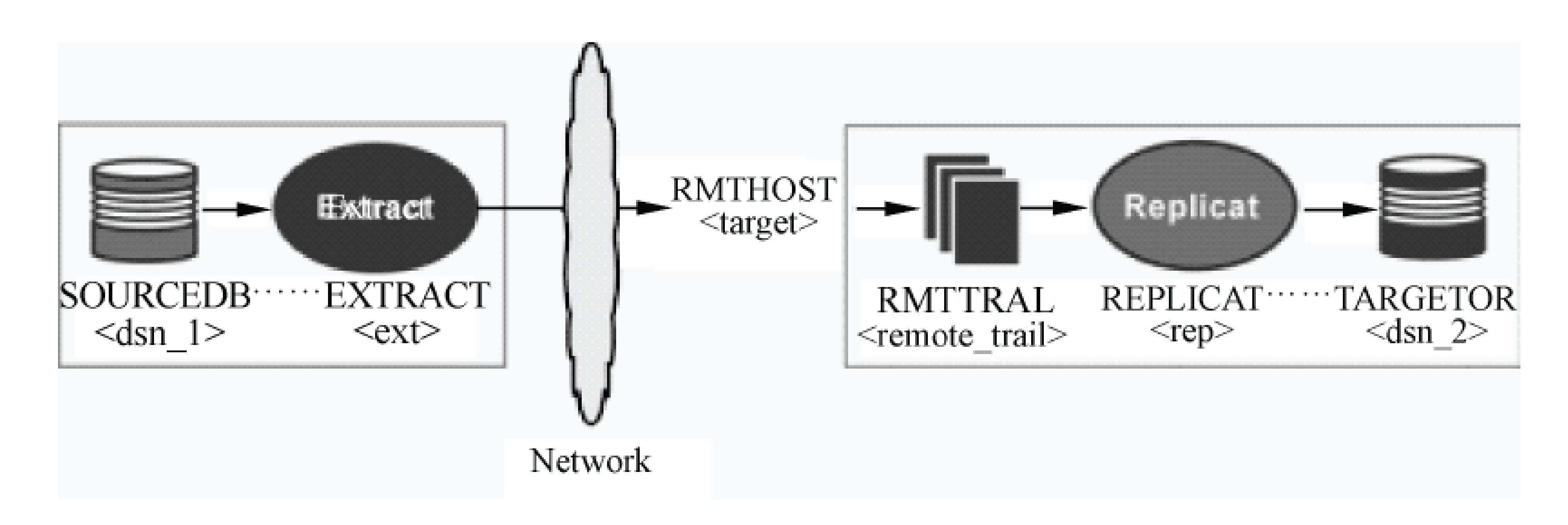


图 2-12

2.3.1　配置源端 MGR 管理进程组

管理进程组在源端与目标端都是必须要有的，它负责启动 GoldenGate 进程，以及相关的动态进程，trail 文件的管理，错误信息报告等。

1. 设置编辑器

Windows 下默认的编辑器就是 notepad，UNIX 下默认的编辑器是 vi。如果对默认的编辑器不熟悉，可以在 GGSCI 下运行 set editor 把编辑器设置为你熟悉的编辑器，例如把默认的编辑器设置为 emacs 时可以使用：

示例 2-14：

```
GGSCI(source) 1>set editor emacs
```

另外 Linux 平台也可以在用户环境变量中设置 EDITOR 参数来选择你熟悉的编辑器。

2. 配置参数文件

进入 GGSCI，edit params mgr，在弹出的文件里添加需要的参数内容。

操作步骤如图 2-13 所示：

```
GGSCI (source) 25> edit params mgr


GGSCI (source) 26> start mgr

Starting Manager as service ('GGSMGR')...
Service started.


GGSCI (source) 27> view params mgr
port 7500
dynamicportlist 7501-7505
autorestart extract *,waitminutes 2, retries 5
```

图 2-13

这里 port 参数表示 MGR 进程通信的端口号，在 MGR 的配置文件中，只有这个参数是必须存在的。dynamicportlist 参数表示 manager 进程可以为源端和目标端的动态通信动态地指定端口，autorestart extract 表示自动重启 Extract 进程组，每次尝试的时间间隔为 5 秒，最多尝试 5 次，如果没有成功则放弃。

2.3.2 配置 Extract 抽取进程组

抽取进程组在源端运行，负责抓取需要传输的数据。

1. 创建和编辑 Extract 进程配置文件

创建一个名字为 eora 的 Extract 进程组，并编辑必要的参数：

示例 2-15：

```
edit params eora
```

然后在弹出的文本文件里添加需要的参数内容，操作如图 2-14 所示：

Extract eora 表示这是一个 Extract 进程，名字为 eora，这里名字需要和 edit params 后

面的名字相对应。dynamicresolution 指的是 GoldenGatex 动态解析源端的表名。与非动态对应，默认 GoldenGate 会在一个进程启动的时候到数据库中查询表的属性，然后创建一个对象记录。这条记录在内存以及磁盘中维护，如果需要复制的表很多，那么创建的过程非常耗时。

```
GGSCI (source) 28> edit params eora


GGSCI (source) 29> view params eora
extract eora
dynamicresolution
userid ggs,password ggs
setenv(ORACLE_SID=shen)
exttrail c:\gg\dirdat\et
table scott.*;
```

图 2-14

另外值得注意的是可以用 setenv 命令来设置 GoldenGate 操作系统的环境变量。

在 GGSCI 中添加 Extract 进程，在源端用 add extract 命令创建一个 Extract 进程：

示例 2-16：

```
add extract  eora, tranlog, begin now
```

用 add exttrail 命令创建本地 trail 文件，Extract 组负责写这部分文件，Pump 进程负责读它，操作如图 2-15 所示：

```
GGSCI (source) 30> add extract eora,tranlog,begin now
EXTRACT added.


GGSCI (source) 31> add exttrail c:\gg\dirdat\et,extract eora
EXTTRAIL added.
```

图 2-15

示例 2-17：

```
add exttrail C:\gg\dirdat\et, extract eora
```

2．使用 GGSCI 命令管理 Extract

在 GGSCI 命令行中，可以使用 start、stop、add、alter、cleanup、delete、info、kill 命令管理 Extract 进程。这些进程管理命令见名知意，例如 start 表示启动进程，stop 表示关闭，add 表示添加，info 表示输出信息等。在本书的后面会逐渐接触到一部分常用的命令。如果不熟悉语法的话，可以使用 help 动作名进程组名来查询具体的语法，例如：

示例 2-18：

```
help add extract
```

或者也可以参考 GoldenGate Reference 手册。

2.3.3 配置 Pump 投递进程组

前面提到过，在源端一个数据 Pump 进程是 secondary Extract 进程组。如果没有 Pump 进程，则 Extract 进程负责把抽取来的数据投递到目标端。但是配置 Pump 进程的好处在于可以保证当网络有故障的时候，能稳定且没有差错地把数据投递到目标端。

1．创建和编辑 Pump 进程配置文件

进入 GGSCI，用“edit parmas pump_so”配置一个名字为 pump_so 的 Pump 进程。

具体操作如下：

示例 2-19：

```
GGSCI (source) 45> edit params pump_so
GGSCI (source) 46> view params pump_so
Pump eora
Dynamicresolution
Userid ggs,password ggs
Setenv(ORACLE_SID=shen)
Exttrail c:\gg\dirdat\et
Table scott.*;
```

注意这里的 Userid。

2．在 GGSCI 中添加 Pump 进程

在 GGSCI 中添加 Pump 进程操作如图 2-16 所示。

```
GGSCI (source) 58> view params eora
extract eora
dynamicresolution
userid ggs,password ggs
setenv(ORACLE_SID=shen)
exttrail c:\gg\dirdat\et
table scott.*;


GGSCI (source) 59> add extract pump_so,exttrailsource c:\gg\dirdat\et
EXTRACT added.


GGSCI (source) 60>  add rmttrail c:\gg\dirdat\pt,extract pump_so
RMTTRAIL added.
```

图 2-16

3．使用 GGSCI 命令管理 Pump 进程

管理 Pump 进程的命令与 Extract 类似，操作如图 2-17 所示。

2.3.4 创建和配置目标端 MGR 管理进程组

依据在源端配置 MGR 的步骤，配置目标端 MGR，操作如图 2-18 所示。

```
GGSCI (source) 61> start pump_so

Sending START request to MANAGER ('GGSMGR') ...
EXTRACT PUMP_SO starting


GGSCI (source) 62> info all

Program     Status      Group       Lag           Time Since Chkpt

MANAGER     RUNNING
EXTRACT     STOPPED     EORA        00:00:00      00:38:08
EXTRACT     STOPPED     PUMP_SO     00:00:00      00:02:18
```

图 2-17

```
GGSCI (target) 21> edit params mgr


GGSCI (target) 22> view params mgr
port 7500
dynamicportlist 7501-7505
autostart er *
autorestart extract *,waitminutes 2,retries 5
lagreporthours 1
laginfominutes 3
lagcriticalminutes 5
purgeoldextracts c:\gg\dirdat\rt*,usecheckpoints, minkeepdays 3


GGSCI (target) 23> start mgr

Starting Manager as service ('GGMGR')...
Service started.


GGSCI (target) 24> info all

Program     Status      Group       Lag           Time Since Chkpt

MANAGER     RUNNING
```

图 2-18

2.3.5　配置 Replicat 复制进程组

Replicat 进程在目标端运行，它负责读源端抽取进程抽取的文件，然后把变化应用到目标端，下面就来配置 Replicat 进程组。

1．创建和编辑 Replicat 进程配置文件

创建和编辑 Replicat 进程配置文件的操作如图 2-19 所示。

这里出现的参数在以后会有进一步的解释。

2．在 GGSCI 中添加 Replicat 并管理

在 GGSCI 中添加 Replicat 并管理的操作如图 2-20 所示。

```
GGSCI (target) 25> edit params rep1


GGSCI (target) 26> view params rep1
replicat rep1
userid ggs, password ggs
assumetargetdefs
reperror default,discard
discardfile ./dirrpt/rep1.dsc, append, megabytes 50
dynamicresolution
map scott.*, target scott.*;
```

图 2-19

```
GGSCI (target) 28> add replicat rep1,exttrail c:\gg\dirdat\pt
REPLICAT added.


GGSCI (target) 29> start rep1

Sending START request to MANAGER ('GGMGR') ...
REPLICAT REP1 starting


GGSCI (target) 30> info all

Program     Status      Group       Lag           Time Since Chkpt

MANAGER     RUNNING
REPLICAT    RUNNING     REP1        00:00:00      00:00:06
```

图 2-20

2.4 验证 DML 复制结果

在源端数据库分别增删改一条数据，到目标端验证是否有数据的增删改，如图 2-21 和图 2-22 所示。

```
SQL> conn scott/tiger
Connected.
SQL> insert into demo values(10,'hello');

1 row created.

SQL> commit;

Commit complete.
```

图 2-21

```
SQL> select  * from demo where id = 10;

        ID ENAME
---------- ----------
        10 hello

SQL> ho echo %computername%
target
```

图 2-22

Insert 数据验证成功如图 2-23 和图 2-24 所示。

```
SQL> conn scott/tiger
Connected.
SQL> update demo set ename = 'updatenew' where id = 10;

1 row updated.

SQL> commit;

Commit complete.

SQL> ho echo %computername%
SOURCE

SQL>
```

图 2-23

```
SQL> select  * from demo where id = 10;

        ID ENAME
---------- ----------
        10 updatenew

SQL> ho echo %computername%
target
```

图 2-24

Update 数据验证成功如图 2-25 和图 2-26 所示。

```
SQL> delete from demo where id = 10;

1 row deleted.

SQL> commit;

Commit complete.

SQL> select * from demo where id = 10;

no rows selected

SQL> ho echo %computername%
SOURCE
```

图 2-25

```
SQL> select  * from demo where id = 10;

no rows selected

SQL> ho echo %computername%
target
```

图 2-26

Delete 数据验证成功。

第 3 章　Linux 平台 Oracle RAC-Oracle Standalone 复制

3.1　目标概述

在 Vmware 虚拟机模拟环境下利用 GoldenGate 实现数据同步，GoldenGate 可以复制 DML 和 DDL 操作。

（1）Oracle RAC 环境见表 3-1。

表 3-1

节点 1	节点 2
操作系统：Red Hat Linux 5.4	操作系统：Red Hat Linux 5.4
数据库：Oracle 10g	数据库：Oracle 10g
IP 地址：	IP 地址：
eth0:192.168.0.101	eth0:192.168.0.102
eth1:10.10.17.201	eth1:10.10.17.202
网关：192.168.0.168	网关：192.168.0.168
DNS：192.168.0.168	DNS：192.168.0.168
内存：512G	内存：512G
网卡：2 块	网卡：2 块
存储管理： Oracle ASM	

（2）Oracle Standalone 环境见表 3-2。

表 3-2

操作系统：	Red Hat Linux 5.4
数据库：	Oracle 10g
IP 地址：	192.168.0.104
网关：	192.168.0.168
DNS：	192.168.0.168
内存：	512G
网卡：	1 块

3.2　GoldenGate 在 Linux 平台的安装

3.2.1　安装前准备工作

首先，我们需要下载 GoldenGate For Linux 版本，比如到 http://edelivery.oracle.com。

1. 创建 GoldenGate 操作系统用户

这里直接使用 Oracle 用户安装 GoldenGate，而不创建新的用户。

2. 准备集群文件系统

这里把 GoldenGate 安装到每一个 RAC 节点，GoldenGate 在每一个节点使用相同的目录结构，然后把 checkpoint 文件和 trail files 放到 ASM 文件系统来实现高可用性。

3. 创建安装目录分配存储空间

/ggs 目录为 GoldenGate 的安装目录。

4. 设置 GoldenGate 用户的环境变量

在两个 RAC 节点，GoldenGate 用户的.bash_profile 文件分别加入，把 GoldenGate 的目录加入到 PATH 中，另外输出 PATH 和 LD_LIBRARY_PATH 这两个环境变量，如图 3-1 所示。

```
export PATH=/u01/ggs/10.0:$PATH
export LD_LIBRARY_PATH=$ORACLE_HOME/lib:/u01/ggs/10.0:$LD_LIBRARY_PATH
```

图 3-1

5. 安装 GoldenGate

把适合操作系统版本的 GoldenGate 上传到每个 RAC 节点，使用 unzip 命令解压，这样得到一个 tar 包如图 3-2 所示。

```
[oracle@node1 10.0]$ unzip V18428-01\(32bit\).zip
Archive:  V18428-01(32bit).zip
  inflating: ggs_redhatAS50_x86_ora10g_32bit_v10.4.0.19_002.tar
```

图 3-2

使用命令 tar -xvof <filename>.tar　解压 GoldenGate 可以看到很多文件如图 3-3 所示。

```
[oracle@node1 10.0]$ tar -xvof ggs_redhatAS50_x86_ora10g_32bit_v10.4.0.
r
mgr
ggsci
ggMessage.dat
help.txt
bcrypt.txt
```

图 3-3

在一个新的目录运行 GGSCI 会出现图 3-4 的界面，这里在/u01/ggs 目录运行。

```
[oracle@node1 ggs]$ ggsci

Oracle GoldenGate Command Interpreter for Oracle
Version 10.4.0.19 Build 002
Linux, x86, 32bit (optimized), Oracle 10 on Sep 25 2009 12:49:31

Copyright (C) 1995, 2009, Oracle and/or its affiliates.  All rights r


GGSCI (node1) 1>
```

图 3-4

输入命令 create subdirs 创建 GoldenGate 的工作目录如图 3-5 所示。

```
GGSCI (node1) 1> create subdirs

Creating subdirectories under current directory /u01/ggs

Parameter files                /u01/ggs/dirprm: created
Report files                   /u01/ggs/dirrpt: created
Checkpoint files               /u01/ggs/dirchk: created
Process status files           /u01/ggs/dirpcs: created
SQL script files               /u01/ggs/dirsql: created
Database definitions files     /u01/ggs/dirdef: created
Extract data files             /u01/ggs/dirdat: created
Temporary files                /u01/ggs/dirtmp: created
Veridata files                 /u01/ggs/dirver: created
Veridata Lock files            /u01/ggs/dirver/lock: created
Veridata Out-Of-Sync files     /u01/ggs/dirver/oos: created
Veridata Out-Of-Sync XML files /u01/ggs/dirver/oosxml: created
Veridata Parameter files       /u01/ggs/dirver/params: created
Veridata Report files          /u01/ggs/dirver/report: created
Veridata Status files          /u01/ggs/dirver/status: created
Veridata Trace files           /u01/ggs/dirver/trace: created
Stdout files                   /u01/ggs/dirout: created
```

图 3-5

6．创建 GoldenGate 数据库用户 ggs 及授 dba 权限

创建 Golden Gate 数据库用户 ggs 及授 dba 权限，如图 3-6 所示。

```
SQL> create user ggs identified by ggs default tablespace users temporary tables
pace temp;

User created.

SQL> grant dba to ggs;

Grant succeeded.

SQL>
```

图 3-6

7. 配置 ASM 的连通性

如果使用 Oracle 的 ASM 作为存储管理软件，那么就需要确保 GoldenGate 能够同样也能连接到 ASM 实例：

（1）确保 ASM 实例已经添加到 tnsnames.ora 文件。

（2）确保监听可以监听到连接 ASM 实例的请求，listener.ora 需要加入如下的内容：

示例 3-1：

```
SID LIST LISTENER DARAN =
(SID_LIST =
(SID_DESC =
(SID_NAME = PLSExtProc)
(ORACLE_HOME = /u01/app/oracle/product/10.0/db_1)
(PROGRAM = extproc)
)
(SID DESC =
(ORACLE_HOME = /u01/app/oracle/product/10.0/db_1)
(SID_NAME = +ASM1)
)
)
```

3.2.2　使用 Oracle clusterware 管理 GoldenGate

1. 添加一个应用程序 VIP 资源

人们可以通过使用 CRS 来管理 GoldenGate 的资源组，并且使用 RAC 的 vip 连接到 GoldenGate，一旦数据库的某一个节点宕掉，Oracle clusterware 将自动切换到另一个可用的节点。

下面是创建应用程序 vip 资源的步骤：

（1）使用安装 GoldenGate 的用户登录。

（2）为 GoldenGate VIP 资源创建一个 profile：

示例 3-2：

```
$ORA_CRS_HOME/bin/crs_profile create ggatevip \
-t application \
-a $ORA CRS HOME/bin/usrvip \
-o oi=eth0,ov=192.168.0.23,on=255.255.255.0
```

命令解析：

$ORA_CRS_HOME	：	是 clusterware 的 HOME 目录。
Ggatevip	：	是你创建的应用程序 vip 的名字。
oi=eth0	：	是制定公用网卡为 eth0。

ov=192.168.0.23　：设置虚拟 IP 为 192.168.0.23。

on=255.255.255.0 ：设置虚拟 IP 的子网掩码为 255.255.255.0，需要和公网 IP 的子网掩码相同。

你还可以设置更多的参数，详情请查看 Oracle clusterware 文档，如图 3-7 所示。

```
[oracle@node1 ~]$ $ORA_CRS_HOME/bin/crs_profile -create ggatevip -t application
-a $ORA_CRS_HOME/bin/usrvip -o oi=eth0,ov=192.168.0.23,on=255.255.255.0
[oracle@node1 ~]$
```

图 3-7

（3）把这个资源注册到 CRS，运行命令：$ORA_CRS_HOME/bin/crs_register ggatevip，如图 3-8 所示。

```
[oracle@node1 ~]$ $ORA_CRS_HOME/bin/crs_register ggatevip
[oracle@node1 ~]$
```

图 3-8

（4）把 vip 的所有权给 root，执行命令，在 root 下执行，[root@node1 bin]# ./crs_setperm ggatevip -o root，如图 3-9 所示。

```
[root@node1 bin]# ./crs_setperm ggatevip -o root
[root@node1 bin]#
```

图 3-9

（5）为 Oracle 用户分配启动这个资源的权限，如图 3-10 所示。

```
[root@node1 bin]# ./crs_setperm ggatevip -u user:oracle:r-x
[root@node1 bin]#
```

图 3-10

（6）通过 Oracle 用户启动这个资源，如图 3-11 所示。

```
[root@node1 bin]# su - oracle
[oracle@node1 ~]$ $ORA_CRS_HOME/bin/crs_start ggatevip
Attempting to start `ggatevip` on member `node2`
Start of `ggatevip` on member `node2` succeeded.
[oracle@node1 ~]$
```

图 3-11

（7）查看这个资源的状态，比如运行状态以及在哪个节点上运行，如图 3-12 所示。

```
[oracle@node1 ~]$ $ORA_CRS_HOME/bin/crs_stat ggatevip -t
Name           Type           Target    State     Host
------------------------------------------------------------
ggatevip       application    ONLINE    ONLINE    node2
[oracle@node1 ~]$
```

图 3-12

2. 创建一个 action 程序

Action 程序必须放在每个节点相同的目录结构下面（也可以放到共享磁盘上），必须可以接受 3 个参数：start、stop 和 check。操作如图 3-13 所示。

- **start 和 stop**　返回 0 成功，1 失败。
- **check**　返回 0 表示 GoldenGate 在运行，1 表示不运行。

```
[oracle@node1 ggs]$ pwd
/u01/ggs
[oracle@node1 ggs]$ ls
10.0     dirdat   dirout   dirprm   dirsql   dirver
dirchk   dirdef   dirpcs   dirrpt   dirtmp   goldengare_action.scr
[oracle@node1 ggs]$
```

图 3-13

下面是这个示例程序的内容：

示例 3-3：

```
#!/bin/sh
#GoldenGate action.scr
GGS_HOME=/ggate/10.0
start_delay_secs=5
#Include the GoldenGate home in the library path to start GGSCI
export LD LIBRARY PATH=${LD LIBRARY PATH}:${GGS HOME}
#Set the oracle home to the database to ensure GoldenGate will get the
#right environment settings to be able to connect to the database
export ORACLE HOME=/u01/oracle/ora111
#check_process validates that a manager process is running at the PID
#that GoldenGate specifies
check_process () {
if ( [ f
"${GGS_HOME}/dirpcs/MGR.pcm" ] )
then
pid=`cut f8
"${GGS HOME}/dirpcs/MGR.pcm"`
if [ ${pid} = `ps e
|grep ${pid} |grep mgr |cut d
" " f1`
]
then
#manager process is running on the PID - exit success
exit 0
else
#if the manager process has a low PID then the cut should use f2
if [ ${pid} = `ps e
```

```
|grep ${pid} |grep mgr |cut d
" " f2`
]
then
#manager process is running on the PID - exit success
exit 0
else
#manager process is not running on the PID
exit 1
fi
fi
else
#manager is not running because there is no PID file
exit 1
fi
}
#call ggsci is a generic routine that executes a GGSCI command
call ggsci () {
ggsci_command=$1
ggsci_output=`${GGS_HOME}/GGSCI << EOF
${ggsci command}
exit
EOF`
}
case $1 in
'start')
call_ggsci 'start manager'
#there is a small delay between issuing the start manager command
#and the process being spawned on the OS - wait before checking
sleep ${start delay secs}
#check whether manager is running and exit accordingly
check process
;;
'stop')
#attempt a clean stop for all nonmanager
processes
call_ggsci 'stop er *'
#ensure everything is stopped
call_ggsci 'stop er *!'
#stop manager without (y/n) confirmation
call_ggsci 'stop manager!'
#exit success
exit 0
;;
'check')
```

```
check process
;;
Esac
```

3．创建一个应用程序 profile

应用程序 profile 是一个键—值对文本文件，可以使用$ORA_CRS_HOME/bin /crs_profile 来创建和操作这个文件。

（1）创建 profile，使用 Oracle 用户执行下面的命令，其操作如图 3-14 所示。

示例 3-4：

```
$ORA CRS HOME/bin/crs profile
-create GoldenGate_app
-t application
-r ggatevip
-a /u01/ggs/GoldenGate action.scr
-o ci=10
```

```
[oracle@node1 ~]$ $ORA_CRS_HOME/bin/crs_profile -create goldengate_app -t applic
ation -r ggatevip -a /u01/ggs/goldengare_action.scr -o ci=10
```

图 3-14

命令解析：

- **-create GoldenGate_app**　应用程序的名字是 GoldenGate_app。
- **-r ggatevip**　ggatevip 必须在 GoldenGate 启动之前运行。
- **-a /u01/ggs/ GoldenGate_action.scr**　指定 action 脚本的位置，在每一个实例必须可用。
- **–o ci=10**　检查的时间间隔设置为 10。

（2）注册应用程序，用 Oracle 用户执行，如图 3-15 所示。

```
[oracle@node1 public]$ $ORA_CRS_HOME/bin/crs_register goldengate_app
[oracle@node1 public]$
```

图 3-15

（3）修改 GoldenGate_app 应用程序的所有权，用 root 用户执行，如图 3-16 所示。

```
[root@node1 bin]# ./crs_setperm goldengate_app -o oracle
[root@node1 bin]#
```

图 3-16

（4）允许 Oracle 用户启动应用程序 GoldenGate_app，如果你的 GoldenGate 用户和 clusterware 是一个用户，则无需执行图 3-17 所示的语句。

```
[root@node1 bin]# ./crs_setperm goldengate_app -u user:oracle:r-x
```

图 3-17

4. 启动应用程序

在 Oracle 用户下，使用命令$ORA_CRS_HOME/bin/crs_start GoldenGate_app，可以启动应用程序，如图 3-18 所示。

```
[oracle@node1 ~]$ $ORA_CRS_HOME/bin/crs_start goldengate_app
```

图 3-18

5. 管理应用程序

在 GoldenGate 运行的时候，你想让 GoldenGate 在另一个服务器运行，你可以使用./crs_relocate -f GoldenGate_app 接口使它强行漂移，如图 3-19 所示。

```
[oracle@node1 ~]$ $ORA_CRS_HOME/bin/crs_relocate -f goldengate_app
```

图 3-19

停止 GoldenGate，使用 Oracle 用户执行，如图 3-20 所示。

```
[oracle@node1 ~]$ $ORA_CRS_HOME/bin/crs_stop goldengate_app
```

图 3-20

6. 清理应用程序

如果你想停止 Oracle clusterware 管理 GoldenGate，并且清理已经注册的资源，需要进行以下操作：

（1）以 Oracle 用户，停止 GoldenGate 应用程序，如图 3-21 所示。

```
[oracle@node1 ~]$ $ORA_CRS_HOME/bin/crs_stop goldengate_app
```

图 3-21

（2）使用 Oracle 用户，停止 vip 资源，如图 3-22 所示。

```
[oracle@node1 ~]$ $ORA_CRS_HOME/bin/crs_stop ggatevip
```

图 3-22

（3）取消注册 GoldenGate_app 应用程序资源，需要用 GoldenGate_app 用户执行（这里使用的是 Oracle），如图 3-23 所示。

```
[oracle@node1 ~]$ $ORA_CRS_HOME/bin/crs_unregister goldengate_app
```

图 3-23

（4）取消注册 VIP 资源（以 root 用户执行），如图 3-24 所示。

```
[root@node1 bin]# pwd
/u01/app/oracle/product/10.2.0/crs_1/bin
[root@node1 bin]# ./crs_unregister ggatevip
```

图 3-24

（5）到你创建 GoldenGate_app 文件的节点，删除文件（以 Oracle 用户执行），如图 3-25 所示。

```
[oracle@node1 ~]$ $ORA_CRS_HOME/bin/crs_profile -delete goldengate_app
```

图 3-25

（6）到你创建 vip 文件的节点，删除文件（以 Oracle 用户执行），如图 3-26 所示。

```
[oracle@node1 ~]$ $ORA_CRS_HOME/bin/crs_profile -delete ggatevip
```

图 3-26

7．使用 ASM 存放 trail 文件注意事项

GoldenGate 支持从 ASM 中捕获数据，但如果使用 ASM 存储重做日志文件和/或者归档日志文件，有一些注意事项。

（1）GoldenGate 需要通过监听连接到 ASM 实例读取日志，所以需要配置监听文件。

（2）为了连接到监听，需要添加连接描述符到 tnsnames.ora 文件

（3）GoldenGate 的 Extract 参数文件必须包括下面的内容：

示例 3-5：

```
TRANLOGOPTIONS ASMUSER <user>@<asm>, ASMPASSWORD <pwd>,ENCRYPTKEY <key>
```

3.2.3　配置源端数据库

由于 GoldenGate 的原理是根据 Oracle 的日志进行抽取复制的，所以 Oracle 的 redo 对于 GoldenGate 非常重要。因为 redo 会频繁地切换，日志组是有限的，并且是循环使用的。

（1）为了让 GoldenGate 不丢失任何的日志，需要打开数据库的归档，在数据库 mount 阶段，使用命令：

示例 3-6：

```
alter database archivelog
```

打开数据库的归档，如果使用了 Flash Recovery Area，系统会自动归档到 db_recovery_file_dest。当然也可以使用以下命令手动指定归档路径：

示例 3-7：

```
alter system set log_archive_dest_1='location=/temp/arch';
```

然后用命令：

示例 3-8：

```
SQL> archive log list
```

可以查看数据库是否归档，如图 3-27 所示。

```
SQL> archive log list
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            +RAC_DISK/racdb/
Oldest online log sequence     24
Next log sequence to archive   25
Current log sequence           25
SQL>
```

图 3-27

（2）GoldenGate 需要数据库开启 supplemental log。GoldenGate 需要准确地知道源端的数据修改了哪些列，需要更为详细的日志，所以需要数据库开启 supplemental log。如果没有开启，则根据下面的操作开启，如图 3-28 所示。

```
SQL> select supplemental_log_data_min from v$database;

SUPPLEME
--------
NO

SQL> alter database add supplemental log data;

Database altered.
```

图 3-28

（3）关闭 recyclebin。这个选项是为了支持 GoldenGate DDL 复制的。关掉 recyclebin 的操作如图 3-29 所示。

```
SQL> alter system set "recyclebin"=off;

System altered.
```

图 3-29

（4）在目标端添加 checkpoint 表。这是为了保证源和目标在传数据的时候不会重复或者少传，添加的步骤为在目标机器上编辑 GOLBALS 文件，添加一行，checkpoint ggs.checkpoint，然后 dblogin 数据库，添加 checkpoint 表，如图 3-30 和图 3-31 所示。

```
GGSCI (linux4) 1> edit params ./GLOBALS


checkpoint ggs.checkpoint
~
```

图 3-30

```
GGSCI (linux4) 2> dblogin userid ggs,password ggs
Successfully logged into database.

GGSCI (linux4) 3> add checkpointtable ggs.checkpoint

Successfully created checkpoint table GGS.CHECKPOINT.

GGSCI (linux4) 4>
```

图 3-31

3.3　配置源端进程组

3.3.1　配置 MGR 进程组

为了编辑和运行 GoldenGate，在源端和目标端都必须运行一个 MGR 进程组，它负责启动 GoldenGate 进程，启动动态进程，管理 trail 文件以及错误信息报告等。

创建动态进程组的步骤如下。

进入安装 GoldenGate 的目录，运行 GGSCI 程序来打开 GoldenGate 命令接口，如图 3-32 所示。

- 在 GGSCI，输入 edit params mgr 命令编辑参数文件。
- 在 GGSCI，输入 port <port_number>指定管理端口。
- 在 GGSCI，还可以配置一些其他的参数，然后输入保存文件。

```
[oracle@node1 10.0]$ ggsci

Oracle GoldenGate Command Interpreter for Oracle
Version 10.4.0.19 Build 002
Linux, x86, 32bit (optimized), Oracle 10 on Sep 25 2009 12:49:31

Copyright (C) 1995, 2009, Oracle and/or its affiliates.  All rights


GGSCI (node1) 1> edit params mgr
```

图 3-32

1．动态端口参数设置

可以制定最大 256 个可用端口的动态列表，当指定的端口不可用时，管理进程将会从列表中选择一个可用的端口，源端和目标端的 Collector、Replicat、GGSCI 进程将会使用这些端口。

指定动态端口的命令：

示例 3-9：

```
DYNAMICPORTLIST {<port> | <port>-<port>} [,…]
```

参数 DYNAMICPORTREASSIGNDELAY <seconds>可以指定多长时间重新制定一次。

2. 自动启动参数设置

当管理进程启动的时候使用 autostart 参数设置 Extract 和 Replicat 进程自动启动，也可以使用 autostart 设置，当 Extract 和 Replicat 进程异常终止时来自动启动。

这个功能在网络临时中断、进程异常终止、数据库死锁等情况下比较有用，它会节省大量的工作量，操作如图 3-33 所示。

```
GGSCI (node1) 2> view params mgr

port 7809
dynamicportlist 7800-8000
autorestart extract *,waitminutes 2,resetminutes 5


GGSCI (node1) 3>
```

图 3-33

下面为这个命令的格式：

示例 3-10：

```
AUTORESTART {ER | EXTRACT | REPLICAT} {group name | wildcard}
[, RETRIES <max retries>]
[, WAITMINUTES <wait minutes>]
[, RESETMINUTES <reset minutes>]
```

这里 autorestart 行表示每 2 分钟尝试重新启动所有的 Extract 进程，一共尝试 2 次（默认 entries 大小为 2，如果需要设置尝试的次数，可以设置 entries 参数），以后 5 分钟清零，然后再按照设 2 分钟尝试一共清理 2 次。

启动 MGR 进程，如图 3-34 所示。

```
GGSCI (node1) 3> start mgr

Manager started.


GGSCI (node1) 4> 
```

图 3-34

3. trail 文件维护

当 GoldenGate 应用完成 trail 文件以后，可以设置定期的自动清理 trail 文件，释放磁

盘空间。在参数文件中使用参数 PURGEOLDEXTRACTS，管理进程就会自动定期的清除 trail 文件。

注 意

默认情况下 GoldenGate 并不会删除过期的 trail 文件，这样，trail 占用的磁盘空间会随着时间的积累越来越大，如不及时清理，会导致对应的文件系统占满，相应的进程会 abend（异常中止），甚至还会引起操作系统的一些问题。

3.3.2 配置 Extract 进程组

抽取进程组在源端运行，负责抓取需要传输的数据。

下面为创建一个名为 eora 的 Extract 进程组。

（1）edit params eora，然后添加需要的参数内容，如图 3-35 所示。

```
GGSCI (node1) 62> view params eora

extract eora
dynamicresolution
userid ggs,password ggs
exttrail /u01/ggs/10.0/dirdat/et
table scott.*;
```

图 3-35

（2）在源端用 add extract 命令创建一个主 Extract 组（因为测试中已添加，所以这里用 alter），如图 3-36 所示。

```
GGSCI (node1) 10> alter extract eora,tranlog,begin now
EXTRACT altered.
```

图 3-36

（3）用 add exttrail 命令创建本地 trail 文件。主 Extract 组负责写这部分文件，Pump 负责把 trail 文件从源端投递到目标端（因为测试中已添加，所以这里用 alter），如图 3-37 所示。

```
GGSCI (node1) 11> alter exttrail /u01/ggs/10.0/dirdat/et,extract eora
EXTTRAIL altered.
```

图 3-37

（4）使用 GGSCI 命令管理 Extract。在 GGSCI 命令中，可以使用 add、alter、cheanup、delete、info、kill 命令管理 Extract 进程，如图 3-38 所示。

```
GGSCI (node1) 19> start extract eora

Sending START request to MANAGER ...
EXTRACT EORA starting
```

图 3-38

3.3.3 配置 Pump 进程组

在源端一个数据 Pump 进程是 secondary Extract 进程组，前面介绍过，如果没有 Pump 进程，则需要配置 Extract 进程把抽取来的数据投递到目标端，但是配置 Pump 进程的好处是可以保证当网络不稳定的时候，能稳定且没有差错的把数据投递到目标端。

下面为创建一个名为 pump_so 的 Pump 进程组。

（1）使用 edit params pump_so，添加内容保存即可，如图 3-39 所示。

```
GGSCI (node1) 63> view params pump_so

extract pump_so
dynamicresolution
passthru
rmthost 192.168.0.104,mgrport 7809,compress
rmttrail /u01/ggs/10.0/dirdat/pt
table scott.*;
--table ggs.test;
```

图 3-39

（2）用 add extract 指定本地 trail 文件（因为测试中已添加，所以这里用 alter），如图 3-40 所示。

```
GGSCI (node1) 18> alter extract pump_so,exttrailsource /u01/ggs/10/0/dirdat/et
EXTRACT altered.
```

图 3-40

（3）用 add rmttrail 指定远程 trail 文件（因为测试中已添加，所以这里用 alter），如图 3-41 所示。

```
GGSCI (node1) 20> alter rmttrail /u01/ggs/10.0/dirdat/pt,extract pump_so
RMTTRAIL altered.
```

图 3-41

3.4 配置目标端进程

3.4.1 配置目标端 MGR 进程组

使用 edit params mgr 编辑 mgr 参数文件然后保存即可，如图 3-42 所示。

```
GGSCI (linux4) 1> view params mgr

port 7809
dynamicportlist 7800-8000
autostart er *
autorestart extract *,waitminutes 2,retries 5
lagreporthours 1
laginfominutes 3
lagcriticalminutes 5
purgeoldextracts /u01/ggs/10.0/rt*,usecheckpoints,minkeepdays 3
```

图 3-42

这里使用了延迟的预警机制：MGR 进程每隔 1 个小时检查 Extract 的延迟情况，如果超过了 3 分钟就把延迟作为信息记录到错误日志中，如果延迟超过了 5 分钟，则把它作为警告写到错误日志中。

3.4.2 配置目标端 Replicat 进程组

Replicat 进程在目标端运行，它负责读源端投递过来的文件数据，然后把变化应用到目标端，下面来配置 Replicat。

（1）使用 edit params 命令创建一个名为 rep1 的参数文件，如图 3-43 所示。

```
GGSCI (linux4) 53> view params rep1

replicat rep1
userid ggs,password ggs
assumetargetdefs
reperror default,discard
discardfile ./dirrpt/rep1.dsc,append,megabytes 50
dynamicresolution
--map ggs.test, target ggs.test;
map scott.*, target scott.*;
```

图 3-43

（2）在目标端用 add replicat 添加一个 rep1 进程（因为测试中已添加，所以这里用 alter），如图 3-44 所示。

```
GGSCI (linux4) 7> alter replicat rep1,exttrail /u01/ggs/10.0/dirdat/pt
REPLICAT altered.
```

图 3-44

3.5 DML 测试

测试按上面的参数配置，是否可以实现数据同步。

（1）首先启动源端的所有需要的进程，如图 3-45 所示。

```
GGSCI (node1) 26> info all

Program     Status      Group       Lag           Time Since Chkpt

MANAGER     RUNNING
EXTRACT     RUNNING     EORA        00:00:00      00:00:08
EXTRACT     RUNNING     PUMP_SO     00:00:00      00:00:08
```

图 3-45

（2）同样启动目标端的所有进程，如图 3-46 所示。

```
GGSCI (linux4) 10> info all

Program     Status      Group       Lag           Time Since Chkpt

MANAGER     RUNNING
REPLICAT    RUNNING     REP1        00:00:00      00:00:01
```

图 3-46

（3）在源端数据库，插入 scott.dept 一行数据，如图 3-47 所示。

```
SQL> select * from dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

SQL> insert into dept values(50,'RENXIAODONG','NIHAO');

1 row created.

SQL> commit;

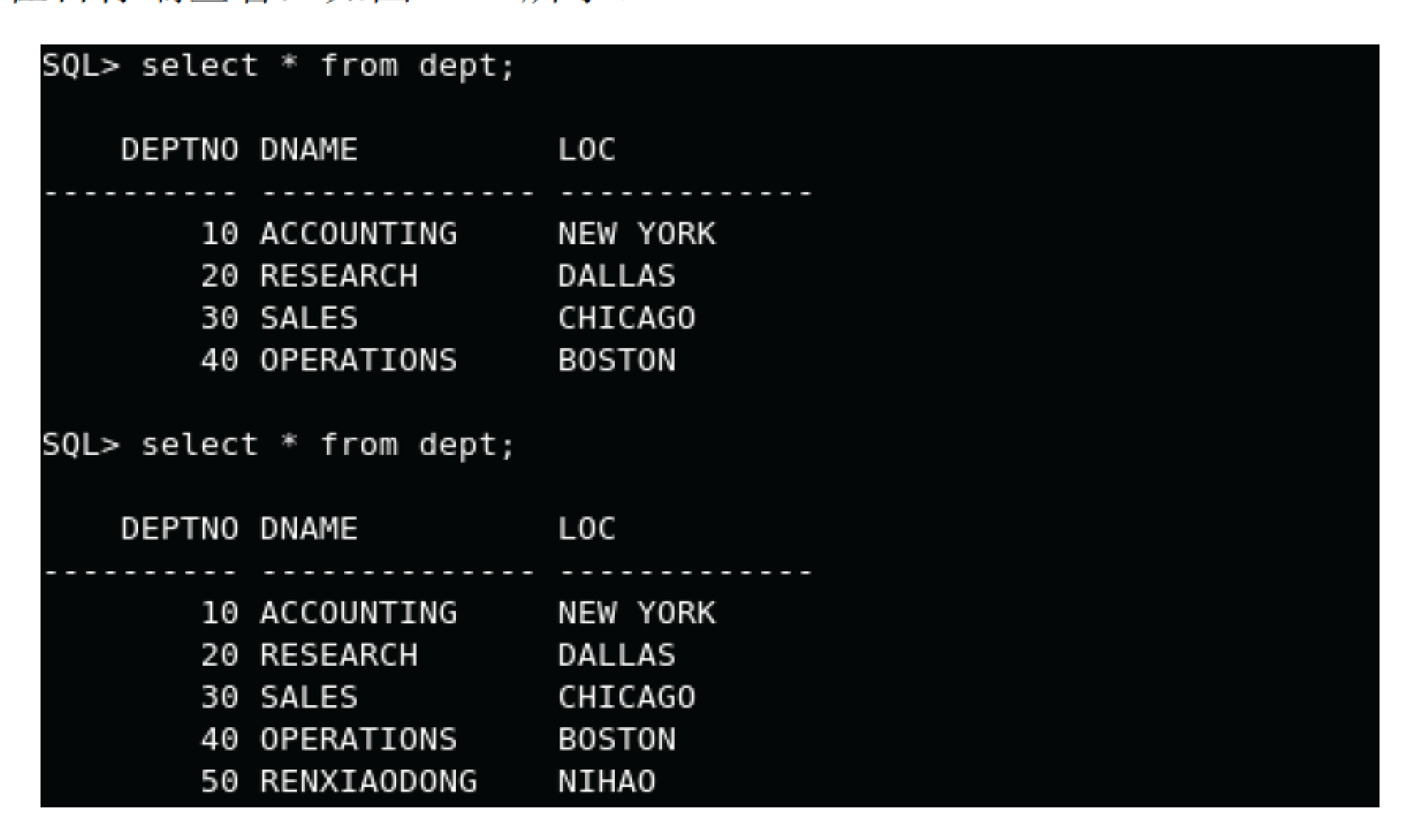
Commit complete.

SQL> select * from dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        50 RENXIAODONG    NIHAO
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

图 3-47

（4）在目标端查看，如图 3-48 所示。



```
SQL> select * from dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

SQL> select * from dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
        50 RENXIAODONG    NIHAO
```

图 3-48

由此可见上面的配置是成功的。

第 2 篇

基　础　篇

第 4 章　目标端数据初始化

前面提到，GoldenGate 作为数据复制的工具，它的原理是将数据从一个数据库容灾到另一个数据库中，由于它是基于交易的复制，是通过读取生产端的 redo 或 archive 日志文件来获取数据的变化，然后在容灾端还原 SQL，从而使生产端和容灾端的数据库数据一致。很多时候在配置 GoldenGate 软件的时候，生产端的数据库中已经存在了大量的数据，那么这些数据就需要同步到容灾端。

数据同步可以在一个活动的生产库上实现，在同步数据的同时，生成库还有数据的更新，这样就需要配置一组进程来实时的获取更新，同步也可以在一个不活动的生产库上实现，当然这样容易得多。

4.1　目标端数据库初始化同步的方法及比较

4.1.1　GoldenGate 初始化数据的方法

GoldenGate 初始化数据一般有以下几种方法。

- 使用数据库工具进行初始化，这种方法最简单最有效。
- Extract 进程把数据抽取到文件然后 Replicat 进程投递到容灾端数据库，这种方法的缺点是比较慢。这种方式相对较简单，但是需要熟悉 ETL 工具。
- Extract 进程把数据抽取到一个外部表，然后通过外部工具导入到容灾库中。Extract 进程通过 TCP/IP 协议直接连接 Replicat 进程，不需要 Collect 进程或文件，通过数据库引擎把数据导入到容灾端，这种方式对网络可靠性以及速度要求比较高。
- Extract 进程把数据抽取到一个外部表文件，然后通过 SQL*Loader 工具把数据导入到容灾端，这种方法比较快。

初始化同步的方法多种多样，但其原理基本相同，都是先把数据库恢复到一个 SCN 或一个时间点以后，然后利用一组抽取和投递进程组来获取数据的变化，最终达到数据同步的目的。

例如利用 Oracle 自带的 RMAN 在线初始化同步数据库，由于这种方法对源端数据库影响较小，并且无需停机进行，而且是基于 SCN 的复制，因此数据一致性保护比较好。与其他的初始化同步方法相比有很多的优势，所以在实际 Oracle-Oracle 容灾项目中，使用这种方式实现初始化同步的较多，另外还有 exp/imp、expdp/impdp、Transportatable tablespace 等，也可以使用其基于 SCN 的导出的方式进行。

具体可以参考 Oracle 数据库方面的知识。



某些工具可以配置多个通道或者并行来提高初始化的速度。

4.1.2　在初始化同步之前需要做的准备

（1）在执行初始化之前，如果配置了 DDL 需要禁用掉，GG 配置是通过在 Extract 和 Replicat 参数文件中配置一下参数来实现同步 DDL 的，所以禁用 DDL 只需要把参数文件中和 DDL 有关的参数去掉就可以了。

（2）如果容灾端已经创建表，需要确保容灾端的表是空的，否则可能导入重复的数据记录使容灾库出现重复数据；禁用表的外键、约束和触发器，外键可能导致错误而约束会减慢插入的速度；删除表的索引，如果表存在索引，数据库在插入数据的时候需要同时更新索引，这会减慢插入的速度。

（3）在容灾端参数文件中加入 HANDLECOLLISIONS 参数来解决数据冲突，但需要表有主键或唯一索引，如果没有主键和唯一索引，则需要在生产端 table 和容灾端 map 配置文件加入 KEYCOLS 参数。

（4）如果生产端和容灾端的数据库不一致，需要配置定义 DEFGEN 文件来实现转换。

（5）配置管理进程。

示例 4-1：

```
edit params mgr

PORT 7839
DYNAMICPORTLIST 7840-7849
--AUTOSTART EXTRACT *
--AUTORESTART EXTRACT *,RETRIES 5,WAITMINUTES 3
PURGEOLDEXTRACTS ./dirdat/*,usecheckpoints, minkeepdays 3
LAGREPORTHOURS 1
LAGINFOMINUTES 30
LAGCRITICALMINUTES 45
```

（6）为了防止在初始化目标端数据库的过程中数据库发生交易数据的变化，需要配置一组在线 Extract 和 Replicat 进程组来获取数据的变化。GoldenGate 只抽取在 Extract 进程启动以后开始的交易。在 Extract 启动之前开始的交易将被跳过，所以在执行初始化之前一定要保证数据库当前的交易都是在 Extract 启动之后开始的。

示例 4-2：

```
-----配置 Extract 进程-----
edit params extya

EXTRACT extya
userid GoldenGate, password GoldenGate
```

```
SETENV （NLS_LANG="AMERICAN_AMERICA.ZHS16GBK"） --此处数据库字符集设为一致
--SETENV （ORACLE SID = "epmsc1"）
GETTRUNCATES
REPORTCOUNT EVERY 1 MINUTES, RATE
numfiles 5000
DISCARDFILE ./dirrpt/extya.dsc,APPEND,MEGABYTES 1000
WARNLONGTRANS 2h,CHECKINTERVAL 3m
EXTTRAIL ./dirdat/ya
TRANLOGOPTIONS  CONVERTUCS2CLOBS
THREADOPTIONS MAXCOMMITPROPAGATIONDELAY 60000
DBOPTIONS ALLOWUNUSEDCOLUMN
TRANLOGOPTIONS altarchivelogdest primary instance epmsc1  /oraarch1 altar-
chivelogdest  instance  epmsc2 /oraarch2
DYNAMICRESOLUTION
table scott.* ;
--------添加 Extract 进程组---------
add extract extya, tranlog, begin now
add exttrail ./dirdat/ya, extract extya, megabytes 500

--------配置 Pump 进程参数
edit params dpeya

EXTRACT dpeya
RMTHOST ip addr , MGRPORT 7839, compress
PASSTHRU
numfiles 50000
RMTTRAIL ./dirdat/ya
DYNAMICRESOLUTION
table scott.* ;
---------添加 Pump 进程组---------
ADD EXTRACT dpeya ,EXTTRAILSOURCE ./dirdat/ya
ADD RMTTRAIL ./dirdat/ya,EXTRACT dpeya,MEGABYTES 500

---------配置 Replicat 进程--------
edit params repya

REPLICAT repya
USERID GoldenGate, PASSWORD GoldenGate
setenv （NLS_LANG="AMERICAN_AMERICA.UTF8"）
--REPORT AT 01:59
REPORTCOUNT EVERY 30 MINUTES, RATE
REPERROR DEFAULT, abend
numfiles 50000
DBOPTIONS ALLOWUNUSEDCOLUMN
MAXTRANSOPS  500000
```

```
GROUPTRANSOPS 10000
CHECKPOINTSECS 40
--HANDLECOLLISIONS
assumetargetdefs
DISCARDFILE ./dirrpt/repya.dsc, APPEND, MEGABYTES 1000
GETTRUNCATES
ALLOWNOOPUPDATES

MAP scott.* , target scott.* ;
--------添加 GLOBALS 参数文件，加入检查电表--------------
Edit params ./GLOBALS
checkpointtable GoldenGate.ckpt

dblogin userid GoldenGate, password GoldenGate
add checkpointtable GoldenGate.ckpt

--------添加 Replicat 进程组----------------------------
ADD REPLICAT repya ,EXTTRAIL ./dirdat/ya  checkpointtable GoldenGate.ckpt
```

4.2 数据库自带工具初始化

图 4-1 为自动工具停机初始化的原理图。

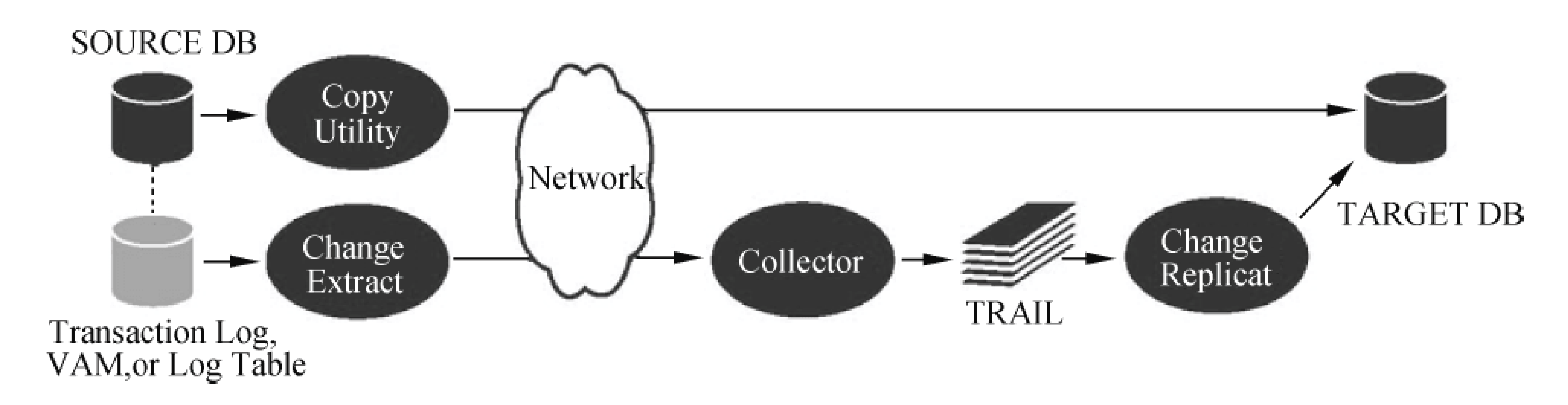


图 4-1

显而易见，这种方法是利用数据库自带的工具 copy 生产端的数据库在容灾端建立数据库。在 copy 数据库的过程中 GoldenGate 启动一组抽取进程来获取在 copy 过程中生产端数据库交易的变化，copy 完成启动容灾端的 Replicat 进程把在 copy 数据库过程中的交易再在容灾端执行一遍。初始化同步完成以后，Extract 和 Replicat 进程持续运行来保证生产端和容灾端数据的一致。

（1）在生产端和容灾 GoldenGate 安装目录下，执行 ./ggsci，然后 start mgr：

示例 4-3：

```
cd /GoldenGate
./ggsci
```

```
GGSCI （target） 1> start mgr
Manager started.
```

（2）启动生产端的 Extract 进程　start extya：

示例 4-4：

```
GGSCI （target） 2> start extya
Sending START request to MANAGER ···
EXTRACT EXTYA starting.
```

（3）拷贝生产库到容灾端直到完成，并记录完成时候的时间。

（4）确认容灾端参数文件中加入了 HANDLECOLLISIONS 参数：

示例 4-5：

```
GGSCI （target） 3>view params repya
REPLICAT repya
USERID ggs , PASSWORD ggs
--SETENV （NLS LANG = "American America.ZHS16GBK"）
--REPORT AT 01:59
REPORTCOUNT EVERY 30 MINUTES, RATE
REPERROR DEFAULT, ABEND
numfiles 50000
DBOPTIONS ALLOWUNUSEDCOLUMN
MAXTRANSOPS  500000
GROUPTRANSOPS 10000
CHECKPOINTSECS 40
HANDLECOLLISIONS
assumetargetdefs
DISCARDFILE ./dirrpt/repya.dsc, APPEND, MEGABYTES 1000
GETTRUNCATES
ALLOWNOOPUPDATES
DDL INCLUDE MAPPED
DDLOPTIONS REPORT
map scott.* , target scott.* ;
```

（5）禁用外键约束和 check 约束，还需要禁用 trigger，因为外键约束可能会导致插入错误，而 check 约束又会减慢插入的速度。数据库中的 trigger 还可能导致冗余数据的产生，例如 GG 插入一份数据，trigger 也同时插入一份数据。

示例 4-6：

```
declare
v_sql varchar2（2000）;
CURSOR c trigger IS SELECT 'alter table '||owner||'.'||table name||'
 disable constraint '||constraint_name
```

```
 from dba_constraints where constraint_type='R' and owner in ('SCOTT',
'TEST');
BEGIN
OPEN c trigger;
LOOP
FETCH c trigger INTO v sql;
EXIT WHEN c trigger%NOTFOUND;
execute immediate v sql;
end loop;
close c trigger;
end;
/


-----------disable references---------
declare
v sql varchar2(2000);
CURSOR c ref IS SELECT 'alter table '||owner||'.'||table name||' disable
constraint '||constraint name from dba constraints where constraint type=
'R' and owner in ('scott','test');
BEGIN
OPEN c trigger;
LOOP
FETCH c ref INTO v sql;
EXIT WHEN c ref%NOTFOUND;
execute immediate v_sql;
end loop;
close c ref;
end;
/
```

（6）启动容灾端的 Replicat 进程，使得在 copy 数据库过程中发生的交易变化在容灾端再重放一遍。

示例 4-7：

```
cd /GoldenGate
./ggsci

GGSCI (target) 2> start repya
Sending START request to MANAGER …
REPLICAT REPYA starting
```

（7）使用命令 info replicat repya 查看 Replicat 进程的时间信息一直到容灾端数据追到 copy 完成的时间点，表明在数据初始化过程中的数据变化已经应用到容灾端。

示例 4-8：

```
GGSCI (target) 2> info repya

REPLICAT   REPYA    Last Started 2011-01-15 13:11   Status RUNNING
Checkpoint Lag      00:00:00 (updated 00:00:15 ago)
Log Read Checkpoint File ./dirdat/ya000002
```

```
                        2011-01-16 21:23:56.938715  RBA 1160096 --查看这个时间
```

（8）由于在 copy 数据库过程中的数据变化已经应用到容灾端，copy 完成这个时间的点容灾端和生产端的数据是一致的，在这个时间点以后只是纯数据的变化，所以不需要冲突处理。使用命令 SEND REPLICAT repay，NOHANDLECOLLISIONS 禁用冲突处理。

示例 4-9：

```
GGSCI （target） 2> SEND REPLICAT repya, NOHANDLECOLLISIONS

Sending NOHANDLECOLLISIONS request to REPLICAT REPYA ···
REPYA No tables found matching GGS.* to set NOHANDLECOLLISIONS
```

（9）去掉容灾端的中的 HANDLECOLLISIONS 参数。

示例 4-10：

```
GGSCI （target） 2> edit params repya

REPLICAT repya
USERID ggs , PASSWORD ggs
--SETENV （NLS LANG = "American America.ZHS16GBK"）
--REPORT AT 01:59
REPORTCOUNT EVERY 30 MINUTES, RATE
REPERROR DEFAULT, ABEND
numfiles 50000
DBOPTIONS ALLOWUNUSEDCOLUMN
MAXTRANSOPS   500000
GROUPTRANSOPS 10000
CHECKPOINTSECS 40
--HANDLECOLLISIONS                    使用--注释掉参数
assumetargetdefs
DISCARDFILE ./dirrpt/repya.dsc, APPEND, MEGABYTES 1000
GETTRUNCATES
ALLOWNOOPUPDATES
DDL INCLUDE MAPPED
DDLOPTIONS REPORT
map scott.* , target scott.* ;
```

（10）初始化完成以后，Extract 进程和 Replicat 进程持续运行来保证生产端和容灾端的数据一致。

4.3 Oracle 的 RMAN 在线初始化

RMAN 在线初始化是使用 RMAN 工具备份生产端的数据库，然后在容灾端恢复数据库到一个 SCN，从这个 SCN 以后启动 Replicat 进程，所以这种方法不需要冲突处理。在备份和恢复的过程中需要启动 Extract 进程组来获取在 backup 和 recover 过程中的数据变化。容灾库恢复完成后启动 Replicat 进程，此后，Extract 和 Replicat 持续运行，保证生产端数

据库和容灾端数据库的一致性。

（1）启动生产端和容灾端的管理进程：

示例 4-11：

```
cd /goldengate
./ggsci

GGSCI （target） 1> start mgr
Manager started.
```

（2）启动生产端的抽取进程，从现在开始获取数据库中变化的事物，然后写到 trail 文件中，在这过程中可以把数据传送到容灾端，但千万不要启动容灾端的 Replicat 进程：

示例 4-12：

```
GGSCI （target） 2> start extya
Sending START request to MANAGER ···
EXTRACT EXTYA starting.
```

（3）查看数据库中所有事务的开始时间，直到其大于抽取进程的启动时间再开始备份数据库，因为 GoldenGate 只获取在 Extract 启动以后的交易变化，在 Extract 启动之前开始而在 Extract 启动以后才完成的交易 GoldenGate 将会忽略这些交易，这些被忽略的交易数据就会丢失。所以需要等数据库中所有的交易都在 Extract 启动之后开始的才能开始备份数据库。通过 v$transaction 视图来查看数据库中的交易：

示例 4-13：

```
SQL>select min (start_time） from v$transaction;
START TIME
--------------------
01/28/11 05:54:14
```

（4）当所有在 Extract 启动之前的开始的交易都完成后，我们就可以使用 RMAN 备份生产端的数据库了。备份数据库的过程中一定要密切监控 Extract 进程的状态，保证其一直正常运行：

示例 4-14：

```
####### rman.sh  begin
export NLS DATE FORMAT='yyyymmdd hh24:mi:ss'
export ORACLE_SID=orcl

rman target / log=/goldnegate/rman.log <<EOF
crosscheck archivelog all;
run{
allocate channel ch1 type disk maxpiecesize 100M;
allocate channel ch2 type disk maxpiecesize 100M;
```

```
backup database tag 'full epmln' format '/nas/backup/%d full %T %U.bak';
sql 'alter system archive log current';
backup archivelog all tag 'arch epmln' format '/nas/backup/%d arch %T %U.bak';

backup current controlfile tag 'ctl_epmln' format '/nas/backup/%d_ctl_%T_%U.bak';
release channel ch1;
release channel ch2;
}
EOF
exit
####### rman.sh  end

--在后台执行：
nohup sh rman.sh &
```

（5）将备份集传送到容灾端。

（6）在容灾端恢复数据库：

示例 4-15：

```
--恢复数据库
startup nomount

--恢复控制文件
run {
allocate channel d1 device type disk;
restore controlfile from 'controlfile backuppiece name';
release channel d1;
}

alter database mount;

--还原数据库
show all
CONFIGURE DEVICE TYPE DISK PARALLELISM 4 BACKUP TYPE TO BACKUPSET;

run {
allocate channel d1 device type disk;
restore database;
release channel d1;
}

--还原归档日志
run {
allocate channel d1 device type disk;
restore archivelog from logseq 34503;
release channel d1;
```

```
}

--恢复数据库
run {
allocate channel d1 device type disk;
recover database using backup controlfile until cancel;
release channel d1;
}

--查询并记录数据文件的 scn
SQL> select CHECKPOINT_CHANGE# ,file# from v$datafile_header;

CHECKPOINT CHANGE# file#
------------------ --------------------
         2493327  1
         2493327  2
         2493327  3            ----一定要记住这个 SCN，启动的 Replicat
--打开数据库（resetlogs）
alter database open database resetlogs
```

（7）修改容灾端数据库为非归档模式：

示例 4-16：

```
--修改参数文件
ALTER SYSTEM RESET log archive dest 1  SCOPE=SPFILE;

--关闭数据库
SHUTDOWN IMMEDIATE

--改为非归档模式
STARTUP MOUNT
ALTER DATABASE NOARCHIVELOG;
ALTER DATABASE OPEN;
```

（8）禁用容灾端数据库的外键，trigger 和有 DML 操作的 JOB，禁用掉容灾端的外键，因为外键约束可能导致插入错误，而 trigger 和有 DML 操作的 JOB 可能导致冗余数据的产生，是容灾端的数据库出现重复数据：

示例 4-17：

```
------disable trigger-------------------
declare
v_sql varchar2（2000）;
CURSOR c trigger IS SELECT 'alter trigger '||owner||'.'||trigger name||'
disable' from dba triggers where owner in （'scott','test'）;
BEGIN
OPEN c_trigger;
```

```
LOOP
FETCH c trigger INTO v sql;
EXIT WHEN c trigger%NOTFOUND;
execute immediate v_sql;
end loop;
close c trigger;
end;
/

-----------disable references---------
declare
v sql varchar2 (2000) ;
CURSOR c ref IS SELECT 'alter table '||owner||'.'||table name||' disable
 constraint '||constraint_name from dba_constraints where constraint_type=
'R' and owner in ('scott','test') ;
BEGIN
OPEN c trigger;
LOOP
FETCH c ref INTO v sql;
EXIT WHEN c_ref%NOTFOUND;
execute immediate v sql;
end loop;
close c ref;
end;
/

---------disable job-----------------

Sql> alter system set JOB QUEUE PROCESSES=0;

--查询数据库运行 scheduler job
col PROGRAM_OWNER for a20
col PROGRAM NAME for a40
col JOB_NAME for a20
set linesize 200 pagesize 100
select job_name,owner,program_name,program_owner,state,enabled
from dba_scheduler_jobs where owner not in ('SYS','SYSTEM') ;

--禁用 scheduler job
exec dbms_scheduler.disable ('EPSA_LN.EPSA_LOG_JOB')
```

（9）启动容灾端的入库进程，一定要根据上面查到的 SCN 启动数据库，因为数据文件恢复到了这个 SCN，由于 RMAN 保证了这个 SCN 之前生产库和容灾库的数据一致性，所以只需要让 GG 来部署这个 SCN 以后出现的数据变化。

示例 4-18：

```
GGSCI 1> start repya , aftercsn xxxxxxxx
```

4.4　GoldenGate initial load 直接传输初始化

GoldenGate 直接传输初始化的原理图，如图 4-2 所示。

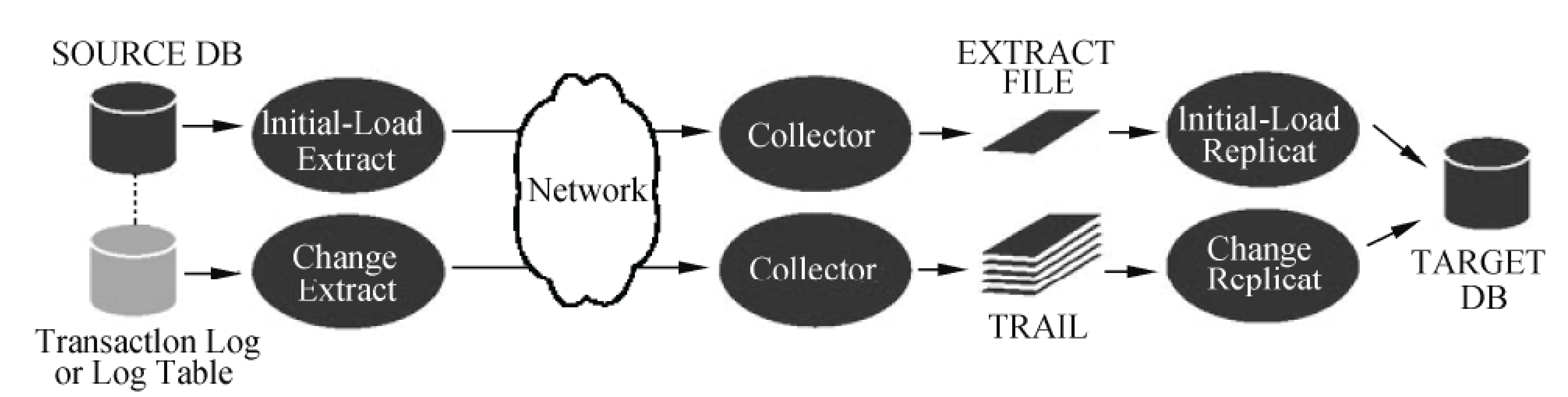


图 4-2

GoldenGate initial load 直接传输初始化是 GoldenGate initial load Extract 进程将获取生产端的记录直接传送给容灾的 initial-load Replicat 进程，这个过程由 MGR 进程动态的启动不需要使用 Collect 进程和文件。

在 initial-load 过程中，需要启动一致 Extract 和 Replicat 进程来获取 initial-load 过程中的数据变化，然后部署到容灾端最终使初始化的结果一致。这种方式不支持 LOB 和 LONG 类型的数据。GoldenGate 会根据 MGR 进程中 DYNAMICPORTLIST 配置的端口列表来为 Replicat 进程动态的分配端口。

在生产端和容灾 GoldenGate 安装目录下，执行 ./ggsci，然后 start mgr：

示例 4-19：

```
cd /goldengate
./ggsci

GGSCI (target) 1> start mgr
Manager started.
```

4.4.1　源端批量抽取的配置

示例 4-20：

```
---------编辑批量抽取参数文件--------------
edit params extinit

EXTRACT extinit
SOURCEDB orcl ,userid ggs , password ggs
RMTHOST ip_addr , MGRPORT 7839, compress
```

```
RMTTASK replicat,GROUP repinit                ----目标端 Replicat
TABLE scott.* ;

---------添加批量 Extract 进程---------------
ADD EXTRACT extinit, SOURCEISTABLE
```

和添加抽取变化进程的方式不同，添加批量抽取进程组的命令为：

示例 4-21：

```
ADD EXTRACT <initial-load Extract name>, SOURCEISTABLE
```

其中，

<initial-load Extract name>：批量抽取进程组的名字，最多八个字符。

SOURCEISTABLE：标识这是一个 initial-load 抽取进程，直接从数据库的表中读取数据。

4.4.2 目标端批量复制的配置

示例 4-22：

```
---------编辑批量复制参数文件--------------
edit params repinit

REPLICAT repinit
TARGETDB orcl,USERID ggs, PASSWORD ggs
ASSUMETARGETDEFS
MAP scott.* , target scott.* ;

---------添加批量 Replicat 进程---------------
ADD REPLICAT repinit, SPECIALRUN
```

和添加抽取变化的命令不同，添加批量复制进程的命令为：

示例 4-23：

```
ADD REPLICAT <initial-load Replicat name>, SPECIALRUN
```

其中：

<initial-load Replicat name>：批量复制进程的名字。

SPECIALRUN：标识批量复制进程组只在一段时间运行，不是永久的 running。

4.4.3 启动批量更新同步

（1）启动生产端的抽取进程 start extya：

示例 4-24：

```
GGSCI （target） 2> start extya
Sending START request to MANAGER ···
```

```
EXTRACT EXTYA starting.
```

（2）启动生产端的批量抽取进程 start extinit，不需要启动 repinit 进程，MGR 会自动启动它，等同步结束，它会自动关闭：

示例 4-25：

```
GGSCI （target） 3> start extinit
Sending START request to MANAGER …
EXTRACT EXTINIT starting.
```

（3）在源端 view report extinti 直到 load 结束，然后做下一步。

（4）在容灾端启动投递进程 start repya：

示例 4-26：

```
GGSCI （target） 1> start repya
Sending START request to MANAGER …
REPLICAT repya starting.
```

（5）查看容灾端投递进程的状态 info repya，直到它大于 load 结束的时间：

示例 4-27：

```
GGSCI （target） 2> info repya

REPLICAT   REPYA     Last Started 2011-01-15 13:11   Status RUNNING
Checkpoint Lag       00:00:00 （updated 00:00:15 ago）
Log Read Checkpoint  File ./dirdat/ya000002
2011-01-16 21:23:56.938715  RBA 1160096    --查看这个时间
```

（6）关掉冲突检查选项 SEND REPLICAT repya，NOHANDLECOLLISIONS：

示例 4-28：

```
GGSCI （target） 3> SEND REPLICAT repya, NOHANDLECOLLISIONS

Sending NOHANDLECOLLISIONS request to REPLICAT REPYA …
REPYA No tables found matching GGS.* to set NOHANDLECOLLISIONS
```

（7）去掉 repya 文件中的 HANDLECOLLISIONS 参数：

示例 4-29：

```
GGSCI （target） 3> edit params repya
--HANDLECOLLISIONS
```

4.5　GoldenGate initial load 使用文件传输初始化

使用文件传输初始化的原理图，如图 4-3 所示。

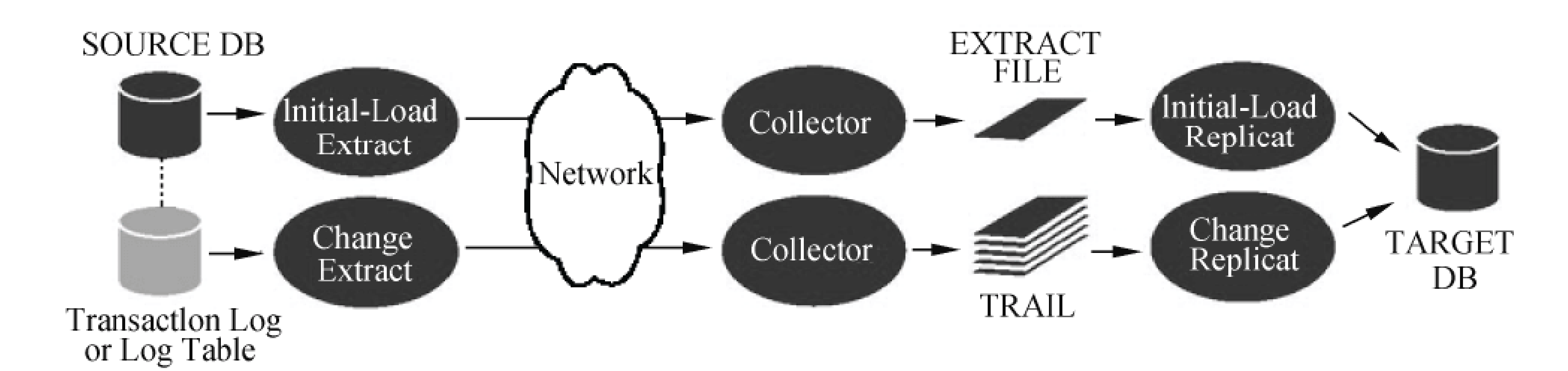


图 4-3

GoldenGate initial-load 使用文件传输初始化的方式是 initial-load Extract 进程直接从源端数据库表中读取数据，然后把它写到一个通用格式的文件中，initial-load Replicat 进程通过数据库接口把文件中的数据复制到容灾端数据库中。

在 initial-load 的过程中 GoldenGate 启动一组 Extract 和 Replicat 进程来获取数据的变化，来保证容灾端数据库和生产端数据库的数据的一致性。

在生产端和容灾 GoldenGate 安装目录下，执行 ./ggsci，然后 start mgr：

示例 4-30：

```
cd /goldengate
./ggsci

GGSCI （target） 1> start mgr
Manager started.
```

4.5.1　配置 initial load extract 进程组

示例 4-31：

```
---------编辑批量抽取参数文件--------------
edit params extinit

SOURCEISTABLE
SOURCEDB orcl ,userid ggs , password ggs
RMTHOST ip addr , MGRPORT 7839, compress
RMTFILE /GoldenGate/rmtfile,MAXFILES 5000, MEGABYTES 500
TABLE scott.* ;
```

这种初始化的方式不需要添加进程组，只需要运行所编辑的文件就可以了。

4.5.2　执行 initial load 捕获进程

（1）在生产端启动抽取变化进程组 start extya。

（2）启动 initial 进程 ./GoldenGate/dirdat/extinit.prm reportfile /GoldenGate/dirrpt/extinit.rpt。

（3）等待直到 initial 完成。

4.5.3　配置 initial load replicat 进程组

示例 4-32：

```
---------编辑批量复制参数文件--------------
edit params repinit

SPECIALRUN
END RUNTIME
EXTFILE /GoldenGate/rmtfile
ASSUMETARGETDEFS
MAP scott.* , target scott.* ;
```

不需要添加 initial load replicat 进程，只需要启动文件就可以。

4.5.4　执行 initial load 复制进程

（1）在容灾端启动 initial load replicat 进程 ./GoldenGate/dirdat/repinit.prm reportfile /GoldenGate/dirdat/repinit.rpt。

（2）等待到 Replicat 完成。

（3）启动 Replicat 进程组 start repya。

（4）查看 Replicat 进程状态，直到其追到 Replicat 完成的时间 info repya。

（5）关闭容灾端的冲突检查操作 SEND REPLICAT <Replicat group name>, NOHANDLECOLLISIONS。

（6）注释掉 Replicat 进程中的 HANDLECOLLISIONS 参数。

第 5 章　为 Oracle 数据库配置 DDL 同步

5.1　不支持及有限支持的 DDL 类型

- Oracle 数据库保留的 SCHEMA、GoldenGate 是不支持复制的：
  ANONYMOUS，AURORA，$JIS，$UTILITY，$AURORA，$ORB，$UNAUTHENTICATED，CTXSYS，DBSNMP，DMSYS，DSSYS，EXFSYS，MDSYS，ODM，ODM_MTR，OLAPSYS，ORDPLUGINS，ORDSYS，OSE$HTTP$ADMIN，OUTLN，PERFSTAT，PUBLIC，REPADMIN，SYS，SYSMAN，SYSTEM，TRACESVR，WKPROXY，WKSYS，WMSYS，XDB。
- GoldenGate10g DDL 不支持 Oracle 回收站功能。
- GoldenGate 支持不超过 2M 长度的 DDL 语句。
- GoldenGate 只支持单向的 DDL 复制，不支持双向的 DDL 复制。
- GoldenGate 只支持源端和目标段结构一致的 DDL 复制和 DDL 有关的 procedure，query 必须保持正确的结构优先执行。

5.2　DDL 处理方法

5.2.1　不支持 DDL 类型的处理方法

不支持 SCHEMA，GG 默认不复制，不需要更多的设置。
关闭 Oracle 回收站：alter system set recyclebin=off scope=both。

5.2.2　受限支持 DDL 类型的处理方法

可以通过脚本 ddl_ddl2file.sql 获取被忽略的 Oracle DDL 操作。它会把获取的操作放在 USER_DUMP_DEST 目录一个文本文件中。

在投递进程中必须使用 ASSUMETARGETDEFS 参数。

5.3　DDL 复制的配置

5.3.1　Oracle DDL 复制的原理

Oracle GoldenGate 的 DDL 复制本质是基于数据库全局 Trigger 的复制。在源库建立一

个 Oracle 全库级的 Trigger 捕捉 DDL 操作到中间表,Extract 读取中间表 DDL 语句并与 DML 语句根据 csn 排序,Pump 投递到目标端，目标端的 Replicat 再重现该 DDL 语句。

DDL 复制与 DML 复制的复制机理是完全不同的，DDL 复制基于 Trigger，而 DML 复制基于日志，两者的复制机理不同，其数据捕捉是没有联系的，只是在主 Extract 进程中通过 scn 号按照发生的顺序进行组装，保证 DDL 操作和 DML 操作按照其原来的顺序执行。

DDL 复制与 DML 复制是相互独立的，DDL 复制的 Trigger 建立和启用后，无论 DML 复制是否运行，该 Trigger 一直在发生作用，捕捉 DDL SQL 语句到中间表。因此，DML 复制的起停并不影响 DDL 的捕获。同样，DDL Trigger 的启用和停止并不影响 DML 复制，只是该 Trigger 被禁止后不再抓取 DDL 操作。它们之间只是在 Extract 进行组装时根据 scn 号进行排序，没有其他任何联系。DDL 复制只是简单的 SQL 复制,通过 Trigger 捕捉 DDL 抓取原始的 SQL 语句并发送到目标重新执行一遍。

5.3.2　安装 GoldenGate DDL 对象

1．停止 Extract 进程组

停止 Extract 进程组如图 5-1 所示。

示例 5-1：

```
Shell> cd <install location>
Shell> ./ggsci
GGSCI> STOP EXTRACT EXT*
```

```
[oracle@node1 ~]$ cd /goldengate
[oracle@node1 goldengate]$ ggsci

Oracle GoldenGate Command Interpreter for Oracle
Version 10.4.0.19 Build 002
Linux, x86, 32bit (optimized), Oracle 10 on Sep 25 2009 12:49:31

Copyright (C) 1995, 2009, Oracle and/or its affiliates.  All rights reserved.


GGSCI (node1) 1> stop ext*

Sending STOP request to EXTRACT EXTYA ...
Request processed.


GGSCI (node1) 2>
```

图 5-1

2．指定源端数据库的模式

DDL 复制的用户用 GoldenGate 的用户来承担，要求权限 DBA：

示例 5-2：

```
Shell> cd <install location>
Shell> ./ggsci
GGSCI> EDIT PARAMS ./GLOBALS
GGSCHEMA  <ddl_schema>---GoldenGate 用户
```

停止 MGR 进程：
示例 5-3：

```
GGSCI> STOP MGR !
```

退出当前 session，如图 5-2 所示。

```
GGSCI (node1) 4> view params ./GLOBALS

ggschema ggs


GGSCI (node1) 5> stop mgr!

Sending STOP request to MANAGER ...
Request processed.
Manager stopped.
```

图 5-2

3．关闭 Oracle 的 recyclebin

由 dba 权限登录数据库：
示例 5-4：

```
Shell> cd <install location>
Shell> sqlplus <ddl_schema>/<password> as SYSDBA
```

检查 recycle bin 是否被关闭：
示例 5-5：

```
SQL> show parameter recyclebin
```

如果没有关闭 recycle bin 用下面命令关闭，如图 5-3 所示：
示例 5-6：

```
SQL> ALTER SYSTEM SET RECYCLEBIN = OFF SCOPE = BOTH;
```

4．安装 DDL 对象

以下所有脚本都在/GoldenGate 目录下执行。
运行 marker_setup 为 GoldenGate DDL 做准备，如图 5-4 所示：

```
[oracle@node1 ~]$ sqlplus /nolog

SQL*Plus: Release 10.2.0.1.0 - Production on Fri Jan 28 04:17:01 2011

Copyright (c) 1982, 2005, Oracle.  All rights reserved.

SQL> conn / as sysdba
Connected.
SQL> show parameter recyclebin

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
recyclebin                           string      ON
SQL> alter system set recyclebin=off scope=both;

System altered.

SQL>
```

图 5-3

```
SQL> @marker_setup.sql

Marker setup script

You will be prompted for the name of a schema for the GoldenGate database object
s.
NOTE: The schema must be created prior to running this script.
NOTE: Stop all DDL replication before starting this installation.

Enter GoldenGate schema name:ggs


Marker setup table script complete, running verification script...
Please enter the name of a schema for the GoldenGate database objects:
Setting schema name to GGS

MARKER TABLE
-------------------------------
OK

MARKER SEQUENCE
-------------------------------
OK

Script complete.
SQL>
```

图 5-4

示例 5-7：

```
SQL> @marker_setup.sql
```

运行 ddl_setup.sql 脚本为 DDL 做准备，如图 5-5 所示：

示例 5-8：

```
SQL> @ddl_setup.sql
```

创建 GoldenGate DDL 复制的角色，如图 5-6 所示：

示例 5-9：

```
SQL> @role_setup.sql
```

```
SQL> @ddl_setup.sql

GoldenGate DDL Replication setup script

Verifying that current user has privileges to install DDL Replication...

You will be prompted for the name of a schema for the GoldenGate database object
s.
NOTE: The schema must be created prior to running this script.
NOTE: On Oracle 10g and up, system recycle bin must be disabled.
NOTE: Stop all DDL replication before starting this installation.

Enter GoldenGate schema name:ggs

You will be prompted for the mode of installation.
To install or reinstall DDL replication, enter INITIALSETUP
To upgrade DDL replication, enter NORMAL
Enter mode of installation:INITIALSETUP

Working, please wait ...
Spooling to file ddl_setup_spool.txt


Using GGS as a GoldenGate schema name, INITIALSETUP as a mode of installation.

Working, please wait ...

RECYCLEBIN must be empty.
This installation will purge RECYCLEBIN for all users.
To proceed, enter yes. To stop installation, enter no.

Enter yes or no:yes
```

图 5-5

```
SQL> @role_setup.sql

GGS Role setup script

This script will drop and recreate the role GGS_GGSUSER_ROLE
To use a different role name, quit this script and then edit the params.sql scri
pt to change the gg_role parameter to the preferred name. (Do not run the script
.)

You will be prompted for the name of a schema for the GoldenGate database object
s.
NOTE: The schema must be created prior to running this script.
NOTE: Stop all DDL replication before starting this installation.

Enter GoldenGate schema name:ggs
Wrote file role_setup_set.txt

PL/SQL procedure successfully completed.


Role setup script complete

Grant this role to each user assigned to the Extract, GGSCI, and Manager process
es, by using the following SQL command:

GRANT GGS_GGSUSER_ROLE TO <loggedUser>

where <loggedUser> is the user assigned to the GoldenGate processes.
SQL>
```

图 5-6

开启 DDL 的功能，如图 5-7 所示：

示例 5-10：

```
SQL> @ddl_enable.sql
```

```
SQL> @ddl_enable.sql

Trigger altered.

SQL>
```

图 5-7

5．验证 DDL 安装

运行脚本 marker_status.sql 校验 DDL 的状态，如图 5-8 所示。

```
SQL> @marker_status.sql
Please enter the name of a schema for the GoldenGate database objects:
ggs
Setting schema name to GGS

MARKER TABLE
-------------------------------
OK

MARKER SEQUENCE
-------------------------------
OK
SQL>
```

图 5-8

5.3.3　配置 DDL 支持

1．停止 Extract 和 Replicat 进程组

源端停止所有的 Extract，如图 5-9 所示。

```
GGSCI (node1) 16> stop extract ext*

Sending STOP request to EXTRACT EXTYA ...
Request processed.
```

图 5-9

目标端停止所有的 Replicat 进程组，如图 5-10 所示。

```
GGSCI (target) 5> stop rep*

Sending STOP request to REPLICAT REPYA ...
Request processed.
```

图 5-10

2．修改 Extract 配置文件

在 Extract 各个进程中将在 table 前加入下面配置，如图 5-11 所示：

示例 5-11：

```
DDL INCLUDE ALL
DDLOPTIONS ADDTRANDATA, REPORT
```

```
DDL INCLUDE ALL
DDLOPTIONS ADDTRANDATA, REPORT

table scott.* ;
sequence scott.* ;
table test.* ;


GGSCI (node1) 22>
```

图 5-11

3．修改 Replicat 配置文件

在 Replicat 各个进程中在 map 前加入下面配置，如图 5-12 所示。

示例 5-12：

```
DDL INCLUDE MAPPED
DDLOPTIONS REPORT
```

```
DDL INCLUDE MAPPED
DDLOPTIONS REPORT
map scott.* , target scott.* ;
map test.gg , target test.gg ;



GGSCI (target) 12>
```

图 5-12

4．重启进程

启动生产端和目标端的进程，如图 5-13 所示。

```
GGSCI (target) 13> start er *

Sending START request to MANAGER ...
REPLICAT REPYA starting
```

图 5-13

5.3.4　验证结果

在生产端执行 DDL 操作，验证 GoldenGate 是否能将 DDL 操作传输到容灾端，在生产端创建表 ddltest，如图 5-14 所示。

```
SQL> conn scott/tiger
Connected.
SQL> create table ddltest(id number,name varchar2(20));

Table created.

SQL> desc ddltest;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 ID                                                 NUMBER
 NAME                                               VARCHAR2(20)

SQL>
```

图 5-14

到目标端查询表 ddltest，如图 5-15 所示。

```
SQL> conn scott/tiger
Connected.
SQL> desc ddltest
 Name                                      Null?    Type
 ----------------------------------------- -------- -------------
 ID                                                 NUMBER
 NAME                                               VARCHAR2(20)

SQL>
```

图 5-15

在目标端查询到了表的机构，可以看到 DDL 复制成功。

5.3.5　DDL 异常与错误处理

处理抽取进程错误如下：

示例 5-13：

```
DDLERROR [, RESTARTSKIP <num skips>] [SKIPTRIGGERERROR <num errors>]
```

加入上述参数跳过一些 DDL 操作来避免抽取进程在没有发现元数据时 ABEND。

处理入库进程错误如下：

示例 5-14：

```
DDLERROR
{<error> | DEFAULT} {<response>}
 [RETRYOP MAXRETRIES <n> [RETRYDELAY <delay>]]
```

```
{INCLUDE <inclusion clause> | EXCLUDE <exclusion clause>}
[IGNOREMISSINGTABLES | ABENDONMISSINGTABLES]
```

利用参数 DDLERROR 处理 GoldenGate 在目标端执行 DDL 错误。

一些 DDLERROR 参数使用的示例如下：

示例 5-15：

```
DDLERROR <error> IGNORE RETRYOP MAXRETRIES 3 RETRYDELAY 10 &
INCLUDE ALL OBJTYPE TABLE OBJNAME "tab*" EXCLUDE OBJNAME "tab1*"
```

在符合通配符的范围，GoldenGate 在出现错误以后重新尝试三次以后忽略规定的操作：

示例 5-16：

```
DDLERROR DEFAULT ABENDS
```

出现错误进程则 GoldenGate 会 ABEND。

处理错误的一般规则就是查看进程的 report 信息和 ggserr.log 查看错误信息，然后分析错误，并解决错误。

示例 5-17：

```
view report 进程名
```

第 6 章　IBM AIX 平台 Sybase-Oracle 数据库复制

6.1　目标概述

本节内容为利用实时数据同步软件 GoldenGate 把数据从 Sybase 数据库（源端）实时复制到 Oracle 数据库（目标端）的实施过程，和以往的试验不同的是这次复制源端和目标端的操作系统、数据库平台都不一样。

部分以前的技术可能无法使用，但是总的原理是一致的。注意本章中对于前期准备工作的细节不再详述，部分请参考前面章节的内容。

复制的原理示意图如图 6-1 所示。

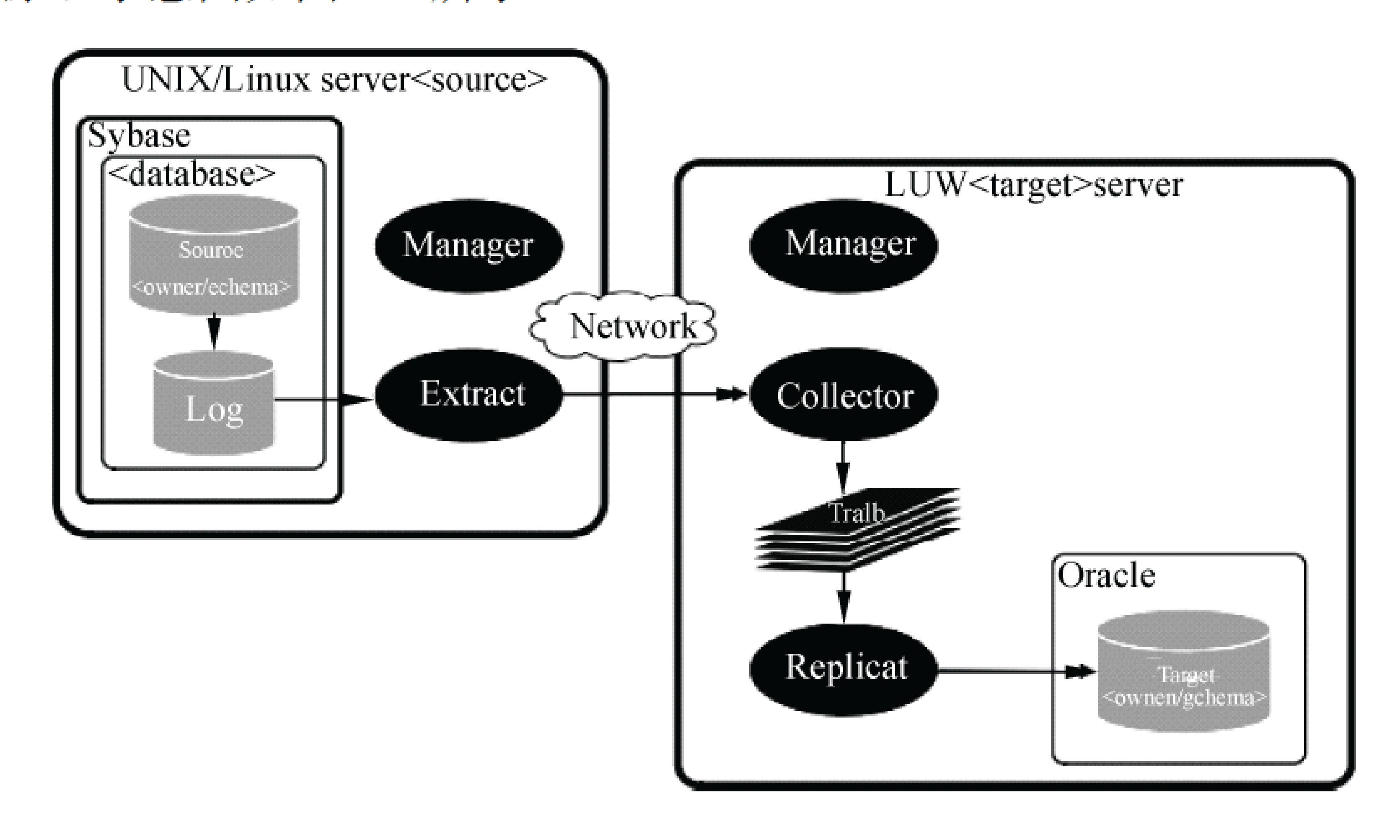


图 6-1

6.2　GoldenGate for sybase 在 AIX 5.3 上的安装注意事项

AIX 5.3 需要 XLC/C++ Runtime v10.1 以上以及 libpthreads version 5.3.0.51 或以上。确认 gg 的环境变量已经设置好，GoldenGate 运行的时候需要用到数据库的一些包，所以需要在 profile 文件里添加相应的 LIBPATH。

步骤概览如下，具体内容如图 6-2 所示。

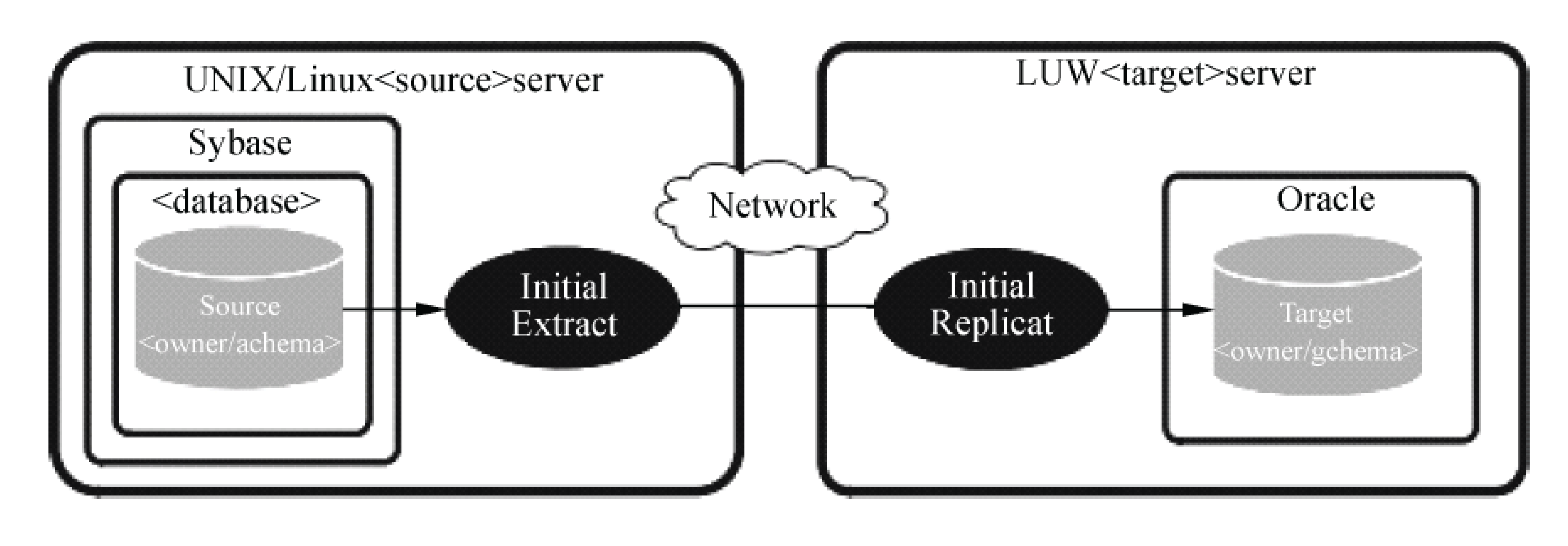


图 6-2

（1）源端和目标端准备好环境，包括准备好数据库，安装好 GoldenGate。
（2）将源端历史数据通过 initial data 初始化到容灾端。
（3）配置源端的抽取，投递进程。
（4）配置容灾端的复制进程。

6.2.1 GoldenGate 在 AIX 操作系统的要求

（1）磁盘空间需求。

依据实际的数据大小，确定 GoldenGate 所需要的实际空间大小，一般需要 50～150M 的空间来安装 GoldenGate 软件，然后根据数据的大小确定队列文件的空间，计算公式为：

```
[log volume in one hour] x [number of hours downtime] x .4 = trail disk space
```

（2）确认/etc/hosts 文件 IP 与 hostname 一一对应。
（3）确认 GoldenGate 使用的端口没有被占用，默认为 7840～7850。
（4）确认安装 GoldenGate 的用户对 GoldenGate 有对应的读写权限。

6.2.2 GoldenGate 对 Sybase 数据库的要求

（1）需要设置 GoldenGate 将用到的环境变量 DSQUERY。
（2）抽取进程将用到 sybase 的 replication API。
（3）抽取进程用 sybase LMT 来读事务日志，所以 RepServer 不可以和 GoldenGate 同运行。
（4）抽取进程必须允许管理 secondary log truncation point。
（5）source replication server 必须处于激活状态，因为 GoldenGate 不能从热备模式的数据库里捕获数据。
（6）源端 sybase 库需要创建一个数据库用户来管理 GoldenGate，并要赋予相应的权限 sp_role 'grant', replication_role, <Extract user>。

此时在源库上简历测试数据，添加表附加日志。

① 在 wei 用户下，hello 数据库上，建立测试数据：

示例 6-1：

```
Shell> isql -Uwei -Pshenweigang -SNODE136 < demo_syb_create.sql
Shell> isql -Uwei -Pshenweigang -SNODE136 < demo_syb_insert.sql
```

② 设置日志截断点：

示例 6-2：

```
[hello@node136gg]$isql -Uwei -Pshenweigang -SNODE136
1>use hello
2>go
1>dbcc settrunc('ltm','valid')
2>go
```

③ 添加表级附加日志：

示例 6-3：

```
GGSCI(node136.com29>dblogin sourcedb hello@NODE136,userid wei,password
shenweigang
Successfully logged into database.
GGSCI(node136.con)30>add trandata dbo.*
Transaction logging is already enanbles for table "dbo"."TCUS TMER".
Transaction logging is already enanbles for table "dbo"."TCUS TORD".
GGSCI(node136.con)31>info trandata*
Transaction logging enanbles for table dbo.TCUS TMER.
Transaction logging enanbles for table dbo.TCUS TORD.
```

6.3　使用 DEFGEN 生成数据表定义文件

利用 DEFGEN 工具可以为源库表或目标端表生成数据定义文件。当源库与目标库类型不一样或者源端的表与目标端的表结构不相同时，数据定义文件是必须要有的。

生成数据定义文件的步骤：

（1）编辑 defgen 文件。

（2）利用 defgen 工具生成 defgen.prm 文件。

（3）将生成好的数据定义文件 ftp 二进制模式传到容灾端对应的目录 dirdef。

6.3.1　编辑 defgen 文件

示例 6-4：

```
GGSCI> edit param defgen
DEFSFILE  dirdef/source.def, PURGE
SOURCEDB hello@NODE136, USERID wei, password shenweigang
```

```
TABLE dbo.TCUSTMER;
TABLE dbo.TCUSTORD;
```

注意，sourcedb 后边的选项 hello 为数据库名，NODE136 为 server 名，一定要跟实际环境一一对应。

6.3.2 利用 defgen 工具生成 defgen.prm 文件

示例 6-5：

```
Shell> defgen paramfile dirprm/defgen.prm
```

生产的数据定义文件内容如图 6-3 所示。

```
[hello@node136 ~]$ cd /gg
[hello@node136 gg]$ cd dirdef
[hello@node136 dirdef]$ ls
source.def
[hello@node136 dirdef]$ cat source.def
*
* Definitions created/modified  2011-02-11 15:47
*
*  Field descriptions for each column entry:
*
*     1     Name
*     2     Data Type
*     3     External Length
*     4     Fetch Offset
*     5     Scale
*     6     Level
*     7     Null
*     8     Bump if Odd
*     9     Internal Length
*    10     Binary Length
*    11     Table Length
*    12     Most Significant DT
*    13     Least Significant DT
*    14     High Precision
*    15     Low Precision
*    16     Elementary Item
*    17     Occurs
*    18     Key Column
*    19     Sub Data Type
```

图 6-3

6.3.3 将生成好的数据定义文件 ftp 二进制模式传到容灾端对应的目录 dirdef

示例 6-6：

```
[oracle@node1 goldengate]$cd dirdef
[oracle@node1 dirdef]$ls
```

```
Source.def
```

6.4　配置源端进程

6.4.1　initial data load

在配置源端进程前，先需要同步初始化数据，这里因为数据库平台不一样，所以不能用传统的同平台数据库相同的方法。一种可行的办法是：把源端数据库导出为文本文件，再导入到目标数据库中。这里采用另外一种方法即 GoldenGate 自带 initial data load 方式。

在源端执行：

示例 6-7：

```
GGSCI> ADD EXTRACT EINIA, SOURCEISTABLE
```

确认添加成功：

示例 6-8：

```
GGSCI> INFO EXTRACT *, TASKS
```

编辑 initial load 抽取进程的参数文件：

示例 6-9：

```
GGSCI> EDIT PARAMS EINIA
EXTRACT EINIA
SOURCEDB hello@NODE136, USERID wei, PASSWORD shenweigang
RMTHOST 192.168.0.137, MGRPORT 7839
RMTTASK REPLICAT, GROUP RINIA
TABLE dbo.TCUSTMER;
TABLE dbo.TCUSTORD;
```

在容灾端也需要配置 RINIA 参数文件：

示例 6-10：

```
GGSCI> EDIT PARAMS RINIA
REPLICAT RINIA
USERID GoldenGate, PASSWORD GoldenGate
DISCARDFILE ./dirrpt/RINIA.dsc, PURGE
SOURCEDEFS ./dirdef/source.def
MAP dbo.TCUSTMER, TARGET SCOTT.TCUSTMER;
MAP dbo.TCUSTORD, TARGET SCOTT.TCUSTORD;
```

此次需要注意数据定义文件为源端生成，且远程传过来的文件：

示例 6-11：

```
GGSCI> ADD REPLICAT RINIA, SPECIALRUN
```

确认已经添加上：

示例 6-12：

```
GGSCI> INFO REPLICAT *, TASKS
```

执行初始化进程：

示例 6-13：

```
GGSCI> START EXTRACT EINIA
```

查看结果：

源端：GGSCI> VIEW REPORT EINIA

示例 6-14：

```
GGSCI (node136.com) 40> view report einia
…………………………………………………………………………………………….
*************************************************************************
**          Running with the following parameters                     **
*************************************************************************
EXTRACT EINIA
SOURCEDB hello@NODE136, USERID wei, PASSWORD ***********
RMTHOST 192.168.0.137, MGRPORT 7839
RMTTASK REPLICAT, GROUP RINIA
TABLE dbo.TCUSTMER;
Using the following key columns for source table dbo.TCUSTMER: CUST CODE.
TABLE dbo.TCUSTORD;
Using the following key columns for source table dbo.TCUSTORD: CUST_CODE,
ORDER DATE,PRODUCT CODE, ORDER ID.
CACHEMGR virtual memory values (may have been adjusted)
………………………………………………………………………………………
Processing table dbo.TCUSTMER

Processing table dbo.TCUSTORD
**************************************************************************
*                ** Run Time Statistics **                               *
**************************************************************************
Report at 2011-02-11 18:15:55 (activity since 2011-02-11 18:15:50)
Output to RINIA:
From Table dbo.TCUSTMER:
      #                 inserts:        2
      #                 updates:        0
      #                 deletes:        0
      #                discards:        0
```

```
From Table dbo.TCUSTORD:
      #                inserts:        2
      #                updates:        0
      #                deletes:        0
      #               discards:        0
```

此处一定要注意，启动 initial 的时候，只需要在源端启动，容灾端会自动复制，确认完成可以用一下命令：

```
目标端：GGSCI> VIEW REPORT RINIA
```

6.4.2　抽取进程与投递进程的配置

1．配置抽取进程 exta

示例 6-15：

```
GGSCI> ADD EXTRACT exta, TRANLOG, BEGIN NOW
GGSCI> ADD EXTTRAIL /gg/dirdat/ga,  EXTRACT exta,  MEGABYTES 25
GGSCI> EDIT PARAM exta
extract exta
setenv (DSQUERY = "NODE136")
sourcedb hello@NODE136, USERID wei, PASSWORD weigang
exttrail /gg/dirdat/ga
DYNAMICRESOLUTION
discardfile ./dirrpt/exta.dsc, purge
TABLE dbo.TCUSTMER;
TABLE dbo.TCUSTORD;
```

这里源库用到的是 hello 这个库，用户名为 wei，传输的表为 demo_syb_create.sql 创建的例子。

查看抽取进程是否添加上：

示例 6-16：

```
GGSCI> info  extract exta
```

2．配置投递进程 dpea

示例 6-17：

```
GGSCI > edit params dpea
extract dpea
passthru
rmthost 192.168.0.137, mgrport 7809
rmttrail /GoldenGate/dirdat/
```

```
discardfile ./dirrpt/dpea.dsc, purge
table dbo.*;
```

这里的容灾端用的是局域网的另一台机器，如图 6-4 所示。

```
[oracle@node1 dirdef]$ hostname
node1
```

图 6-4

6.5 配置目标端进程

6.5.1 在容灾端配置管理进程 MGR

示例 6-18：

```
PORT 7809
DYNAMICPORTLIST 7840-7849
AUTOSTART EXTRACT *
AUTORESTART EXTRACT *,RETRIES 5,WAITMINUTES 3
PURGEOLDEXTRACTS ./dirdat/*,usecheckpoints, minkeepdays 3
LAGREPORTHOURS 1
LAGINFOMINUTES 30
LAGCRITICALMINUTES 45
```

6.5.2 配置全局参数

示例 6-19：

```
Shell> GGSCI
GGSCI> EDIT PARAMS ./GLOBALS
CHECKPOINTTABLE GoldenGate.ggschkpt
GGSCI> EXIT
```

6.5.3 添加检查点表

示例 6-20：

```
GGSCI (node1) 1> dblogin userid GoldenGate, password GoldenGate
Successfully logged into database.
GGSCI (node1) 2> ADD CHECKPOINTTABLE
```

6.5.4 编辑复制进程 repa

示例 6-21：

```
GGSCI (node1) 12> edit param repa
REPLICAT repa
USERID GoldenGate, PASSWORD GoldenGate
SOURCEDEFS ./dirdef/source.def
DISCARDFILE ./dirrpt/eora.DSC, PURGE
MAP dbo.TCUSTMER, TARGET scott.tcustmer;
MAP dbo.TCUSTORD, TARGET scott.tcustord;
```

编辑复制进程 repa 如图 6-5 所示。

```
GGSCI (node1) 12> info all

Program     Status      Group       Lag           Time Since Chkpt

MANAGER     RUNNING
REPLICAT    RUNNING     REPA        00:00:00      00:00:01
```

图 6-5

第 7 章　实际应用中常见场景及案例分析

7.1　目标概述

本章介绍 GoldenGate 常见的场景及一些案例的配置方法。

在生产环境中很多企业对生产数据会有各种各样的需要，如数据仓库，报表系统，容灾等。如果把所有的应用都在一个生产库上实现，势必会给生产库带来很多的压力。为了满足客户多样的需求，GoldenGate 有与之对应的灵活的解决方案，例如可以对数据进行过滤转换，也可以把多个源库的数据汇总到一个数据库。

GoldenGate 的配置相当的灵活。

灵活的拓扑结构如图 7-1 所示。

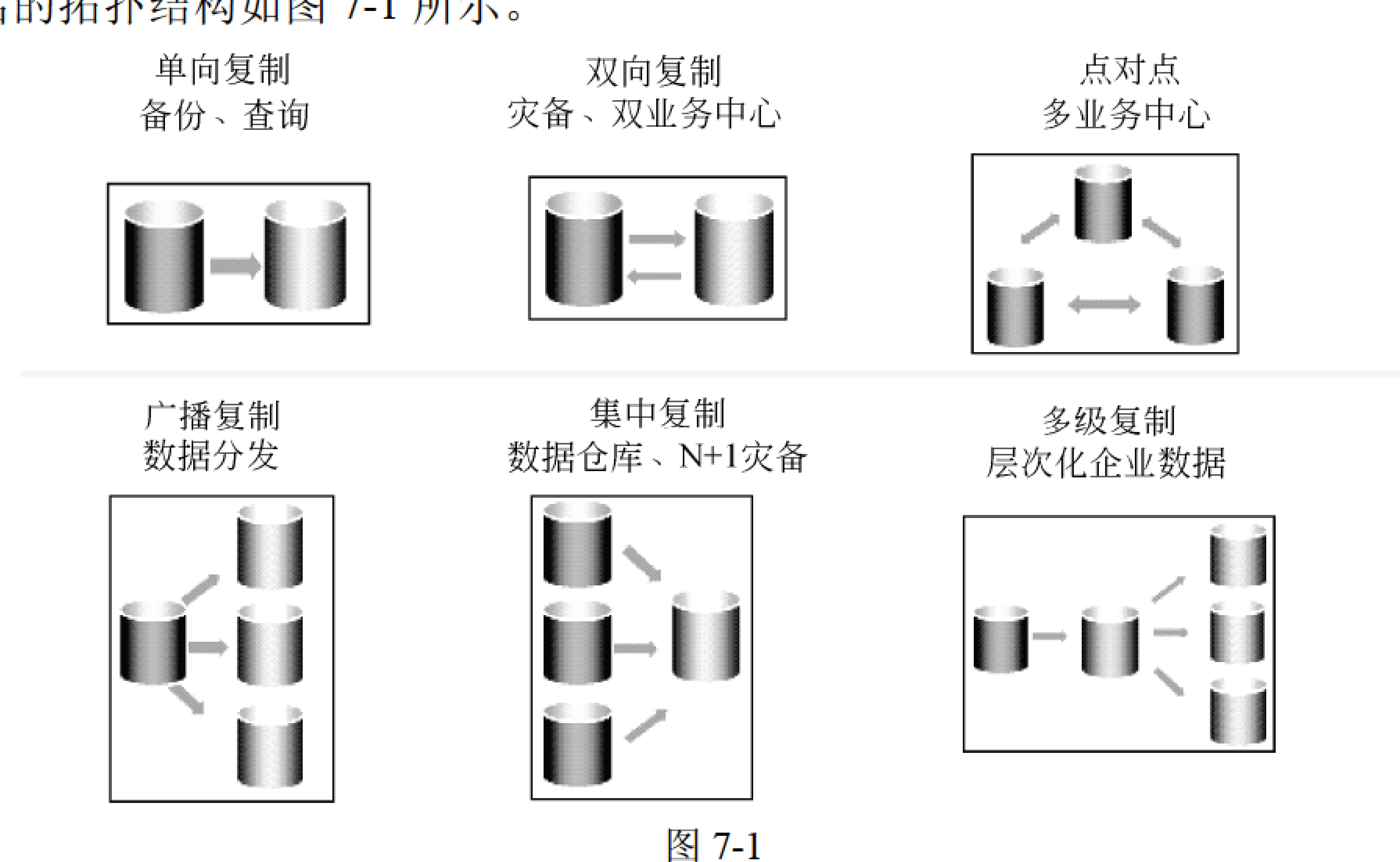


图 7-1

7.2　一对多复制

GoldenGate 支持把一个数据库中的数据同时复制到多个数据库中。这种方式可以让每个系统都得到同样的数据（多对多分数据容灾），也可以使用过滤操作让每个系统只复制用户希望的数据（报表系统，只把用户需要的数据复制过去）。操作如图 7-2 所示。

7.2.1　单源到多目标复制

配置单源到多目标可以有两种方法：独立 Extract 模式和共享 Extract 模式。

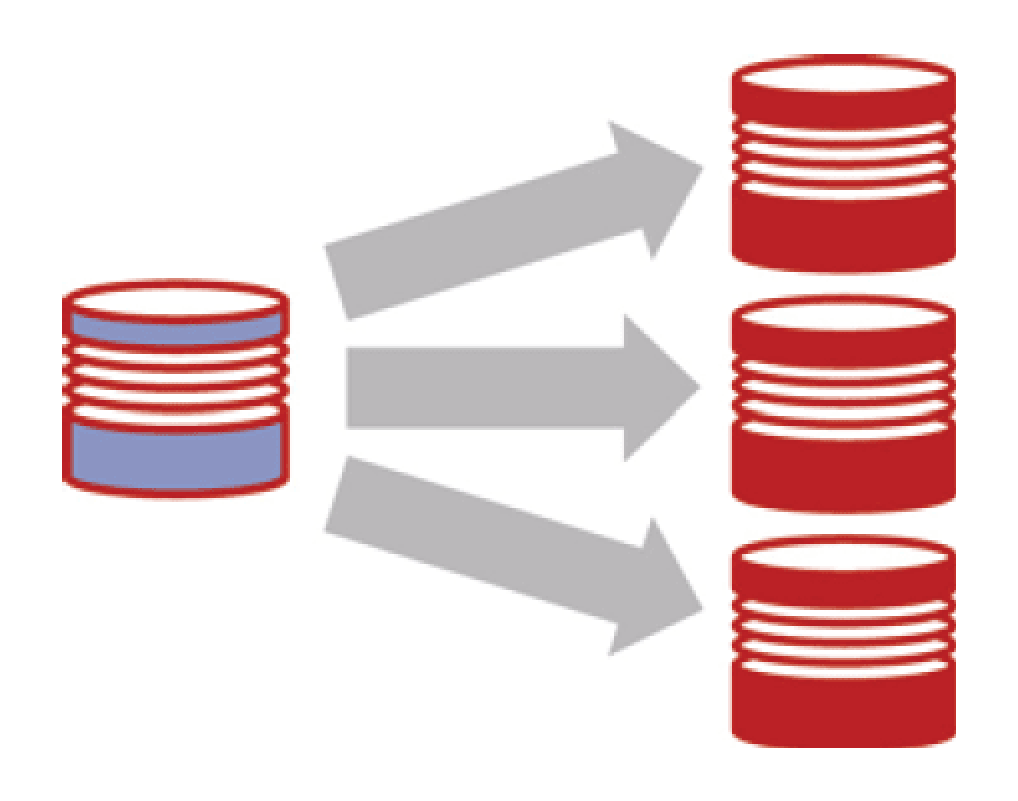


图 7-2

1. 独立 Extract 模式

源端针对每个目标库配置独立的 Extract，如图 7-3 所示。

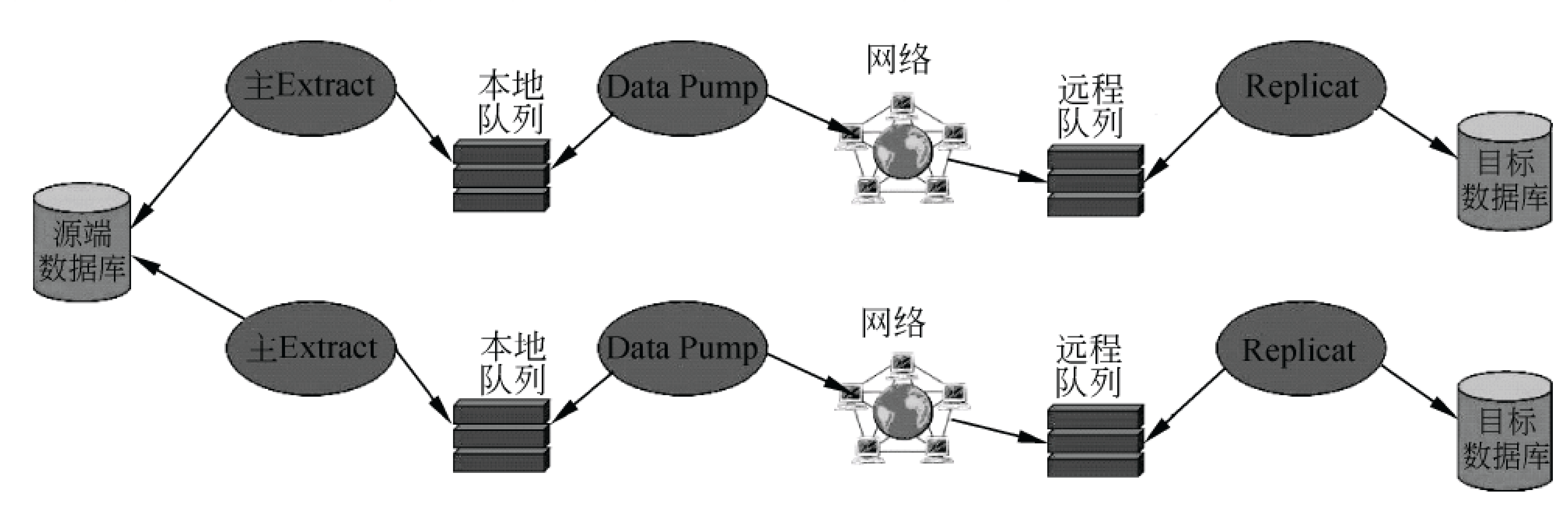


图 7-3

优势是各复制通道无耦合，互相独立，一条链路抽取链路的中断不会影响其他链路的复制，可独立管理数据库也会产生日志非常大的情况。

劣势是需要多个队列，如数据集有重复会占用较多磁盘空间，同时多个 Extract 对于主机资源占用较多，适用于各目标所需数据集无重合或者重合较小并且主机资源充裕的场景。

这种方式是几条完整的 Extract、Pump、Replicat 链路的结合，互不影响，这里介绍一条完整链路的配置方法，其他链路的配置完全一样。下面是一组 Extract、Pump、Replicat 典型的参数配置，其他方式只要略做修改，就可以使用。

生产端和容灾端创建 MGR 进程：

示例 7-1：

```
GGSCI (OE5) 1>edit params mgr

PORT 7839
DYNAMICPORTLIST 7840-7849
--AUTOSTART EXTRACT *
--AUTORESTART EXTRACT *,RETRIES 5,WAITMINUTES 3
PURGEOLDEXTRACTS ./dirdat/*,usecheckpoints, minkeepdays 100
LAGREPORTHOURS 1
```

```
LAGINFOMINUTES 30
LAGCRITICALMINUTES 45
```

生产端配置 Extract 进程参数：

示例 7-2：

```
GGSCI (OE5) 2> edit params extma

EXTRACT extma
userid GoldenGate, password GoldenGate
setenv (NLS LANG="AMERICAN AMERICA.ZHS16GBK")
GETTRUNCATES
REPORTCOUNT EVERY 1 MINUTES, RATE
numfiles 50000
DISCARDFILE ./dirrpt/extma.dsc,APPEND,MEGABYTES 50
WARNLONGTRANS 2h,CHECKINTERVAL 3m
EXTTRAIL ./dirdat/ma
DBOPTIONS ALLOWUNUSEDCOLUMN
TRANLOGOPTIONS  CONVERTUCS2CLOBS
DYNAMICRESOLUTION
TABLE scott.* ;
SEQUENCE scott.*;
```

生产端添加 Extract 进程组：

示例 7-3：

```
GGSCI (OE5) 3>add extract extma, tranlog, begin now
GGSCI (OE5) 4>add exttrail ./dirdat/ma, extract extma, megabytes 500
```

生产端配置 Pump 进程参数：

示例 7-4：

```
GGSCI (OE5) 5>edit params dpema

EXTRACT dpema
RMTHOST ip addr , MGRPORT 7839, compress
PASSTHRU
numfiles 50000
RMTTRAIL ./dirdat/ma
DYNAMICRESOLUTION
TABLE scott.* ;
SEQUENCE scott.* ;
```

生产端添加 Pump 进程组：

示例 7-5：

```
GGSCI (OE5) 6> ADD EXTRACT dpema ,EXTTRAILSOURCE ./dirdat/ma
```

```
GGSCI (OE5) 7> ADD RMTTRAIL ./dirdat/ma,EXTRACT dpema,MEGABYTES 500
```

容灾端配置 Replicat 进程参数：
示例 7-6：

```
GGSCI (OE5) 2>edit params repma
REPLICAT repma
USERID GoldenGate, PASSWORD GoldenGate
setenv (NLS LANG="AMERICAN AMERICA.ZHS16GBK")
--REPORT AT 01:59
REPORTCOUNT EVERY 30 MINUTES, RATE
REPERROR DEFAULT, abend
numfiles 50000
DBOPTIONS ALLOWUNUSEDCOLUMN
MAXTRANSOPS  500000
GROUPTRANSOPS 10000
CHECKPOINTSECS 40
--HANDLECOLLISIONS
assumetargetdefs
DISCARDFILE ./dirrpt/repma.dsc, APPEND, MEGABYTES 1000
GETTRUNCATES
ALLOWNOOPUPDATES
MAP scott.* , TARGET scott.* ;
```

容灾端添加 checkpoint 表：
示例 7-7：

```
GGSCI (OE5) 5>edit params ./GLOBALS

Edit params ./GLOBALS
checkpointtable GoldenGate.ckpt

dblogin userid GoldenGate, password GoldenGate
add checkpointtable GoldenGate.ckpt
```

容灾端添加 Replicat 进程组：
示例 7-8：

```
GGSCI (OE5) 6> ADD REPLICAT repma ,EXTTRAIL ./dirdat/ma  checkpointtable
GoldenGate.ckpt
```

2. 共享 Extract 模式

源端对多个目标配置统一的 Extract，然后通过不同 Data Pump 分发数据，如图 7-4 所示。

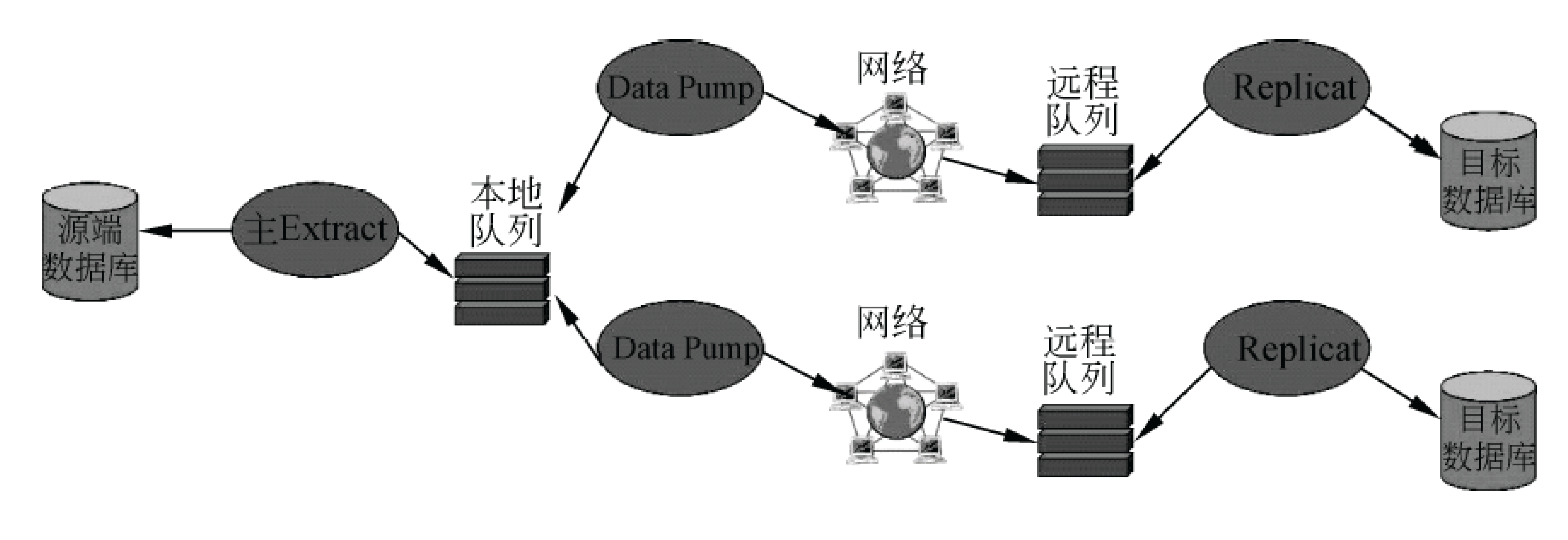


图 7-4

优势是只需一个本地队列，占用空间少，且单个 Extract 占用主机资源少。

劣势是 Extract 作为共享组件出现问题后所有复制均中断。

它适用于多个目标所需数据集相同或有较大重合的场景，对于单源到多目标的复制，主要通常使用 Data Pump 进行数据分发。

共享 Extract 的配置方法如下：

（1）生产端和容灾端配置 MGR 进程参数。

（2）在生产端添加 Extract 进程组和本地 trail 文件：

示例 7-9：

```
ADD EXTRACT <ext>, TRANLOG, BEGIN <time>, [, THREADS]
ADD EXTTRAIL <local_trail>, EXTRACT <ext>
```

（3）编辑 Extract 进程组参数文件，这只是最简单的参数，可以根据上个案例进行详细配置：

示例 7-10：

```
EXTRACT <ext>
[SOURCEDB <dsn_1>,][USERID <user>[, PASSWORD <pw>]]
EXTTRAIL <local_trail>
TABLE <owner>.<table>;
```

（4）添加两个 Pump 进程组，这是这个案例的关键，通过两个 Pump 进程进行数据转发：

示例 7-11：

```
ADD EXTRACT <pump 1>, EXTTRAILSOURCE <local trail>, BEGIN <time>
ADD EXTRACT <pump_2>, EXTTRAILSOURCE <local_trail>, BEGIN <time>
```

（5）添加两个远端 trail 队列：

示例 7-12：

```
ADD RMTTRAIL <remote_trail_1>, EXTRACT <pump_1>
ADD RMTTRAIL <remote_trail_2>, EXTRACT <pump_2>
```

（6）配置两个 Pump 进程组的参数文件：

示例 7-13：

```
--------Pump 1----------
EXTRACT <pump 1>
[SOURCEDB <dsn_1>,][USERID <user>[, PASSWORD <pw>]]
RMTHOST <target_1>, MGRPORT <portnumber>
RMTTRAIL <remote trail 1>
[PASSTHRU | NOPASSTHRU]
TABLE <owner>.<table>;
--------pump 2----------
EXTRACT <pump 2>
[SOURCEDB <dsn_1>,][USERID <user>[, PASSWORD <pw>]]
RMTHOST <target_2>, MGRPORT <portnumber>
RMTTRAIL <remote trail 2>
[PASSTHRU | NOPASSTHRU]
TABLE <owner>.<table>;
```

（7）在容灾端添加两个 Replicat 进程：

示例 7-14：

```
ADD REPLICAT <rep 1>, EXTTRAIL <remote trail 1>, BEGIN <time>
ADD REPLICAT <rep_2>, EXTTRAIL <remote_trail_2>, BEGIN <time>
```

（8）配置容灾端每个 Replicat 的进程参数：

示例 7-15：

```
--------Replicat 1----------
REPLICAT <rep_1>
SOURCEDEFS <full_pathname> | ASSUMETARGETDEFS
[TARGETDB <dsn 2>,] [USERID <user id>[, PASSWORD <pw>]]
REPERROR (<error>, <response>)
MAP <owner>.<table>, TARGET <owner>.<table>[, DEF <template name>];
--------Replicat 2----------
REPLICAT <rep 2>
SOURCEDEFS <full pathname> | ASSUMETARGETDEFS
[TARGETDB <dsn_3>,] [USERID <user id>[, PASSWORD <pw>]]
REPERROR (<error>, <response>)
MAP <owner>.<table>, TARGET <owner>.<table>[, DEF <template name>];
```

7.2.2　单表到多表复制

单表到多表的复制和单源到多目标的复制非常的相似。其中单源到多目标数据库的复制，看其内部就是许多单表到多目标的复制。单表到多表的复制在任何模式的数据库复制中都可以配置，只需要在容灾端配置 MAP 参数即可。

单表到多表的模式：

- 单表到相同（不同）数据库相同 SCHEMA 下不同表名的复制。

- 单表到相同（不同）数据库不同 SCHEMA 下相同表名的复制。
- 单表到相同（不同）数据库不同 SCHEMA 下不同表名的复制。

下面的参数配置是假定把一张表复制到了 3 个表中的配置方法：

示例 7-16：

```
MAP lifeman.gg pol ben , TARGET ELISDATA.pol ben;
MAP lifeman.gg pol ben , TARGET ELISDATA.POL INSURED , colmap ( usedefaults,
CLIENTNO = INSNO);
MAP lifeman.gg_pol_ben , TARGET ELISDATA.POL_JNT_INSURED, colmap
(usedefaults, CLIENTNO = JNT_INSNO);
```

本例中所有目标表主键与源表主键相同，如不同则需使用 keycols 指定。

单表拆分到单表（多表）的复制：

示例 7-17：

```
Replicat 1:
MAP sales.acct, TARGET sales.acct, FILTER (@RANGE (1, 2));
Replicat 2:
MAP sales.acct, TARGET sales.acct, FILTER (@RANGE (2, 2));
```

7.3　多对一复制

多对一的复制一般用在数据仓库中，把多个数据库中的数据复制到一个数据库中。如果不是所有的数据需要复制到数据仓库中，GoldenGate 在复制过程中可以实现过滤、转换和覆盖等操作，只复制有效的数据到数据仓库中，如图 7-5 所示。

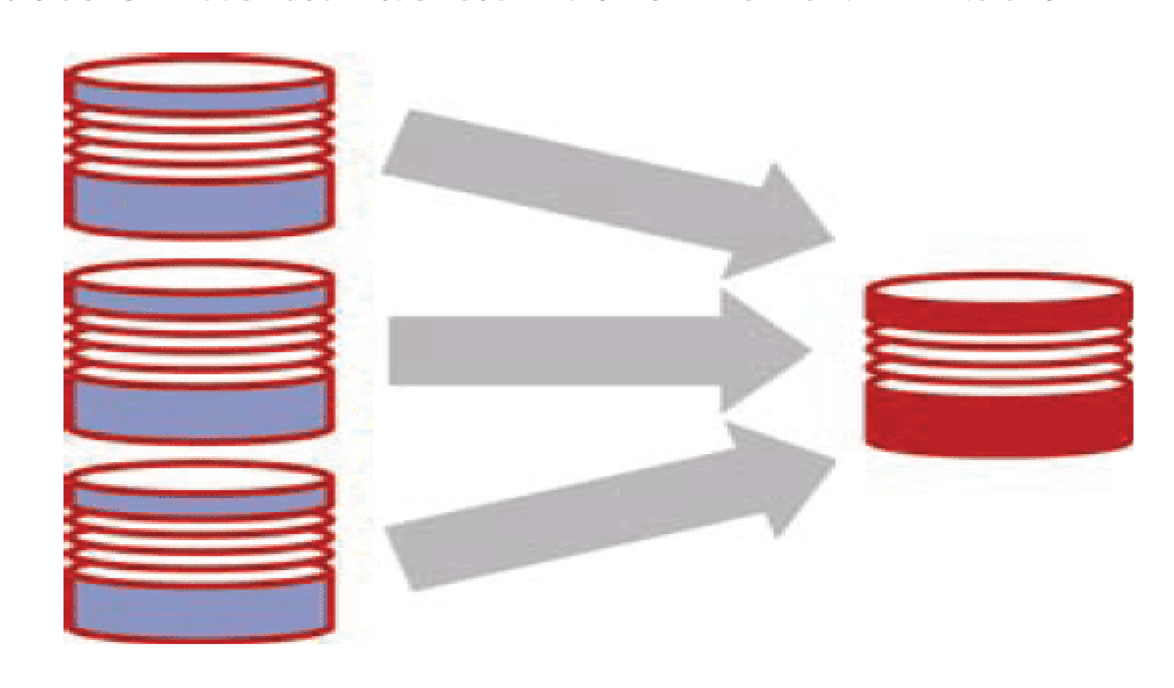


图 7-5

这种多对一的复制假定每个源端数据库复制到目标端数据库的记录都不一样。如果在多个数据库相同表中存在相同的记录，那么在这些数据复制到目标数据库中的时候，就会出现数据冲突，那么就需要通过一些方法来解决数据冲突问题。

多对一的典型案例，比如某大型银行，超过 5 800 家分行及 16 000 多部 ATM，热备份数据中心从跨度很大的不同地区的 4 台主机（HP NSK）抽取数据保障 ATM 7×24 小时可

用，如图 7-6 和图 7-7 所示。

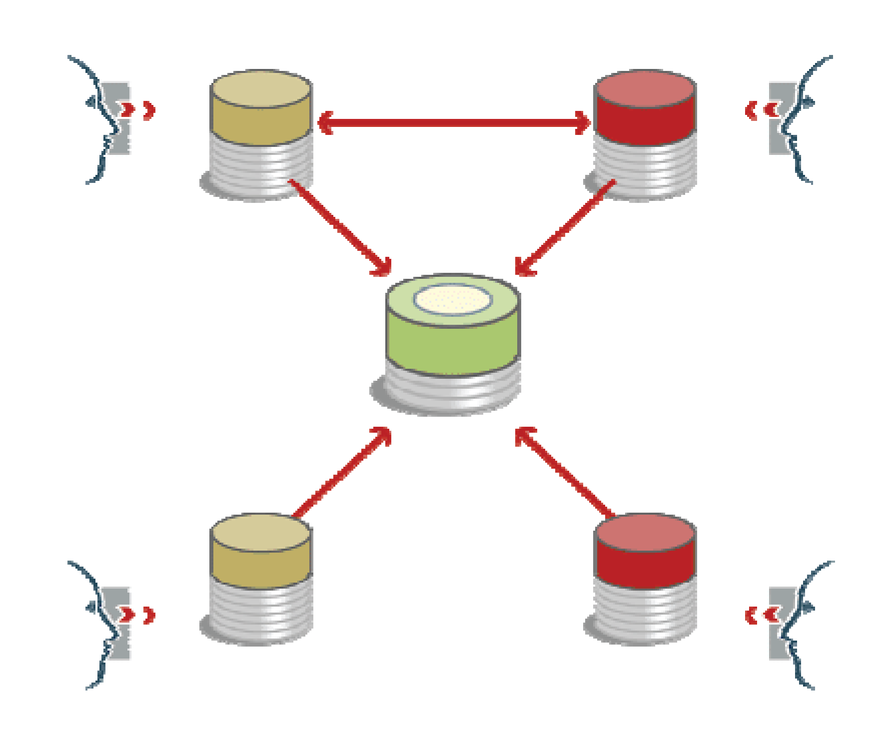


图 7-6

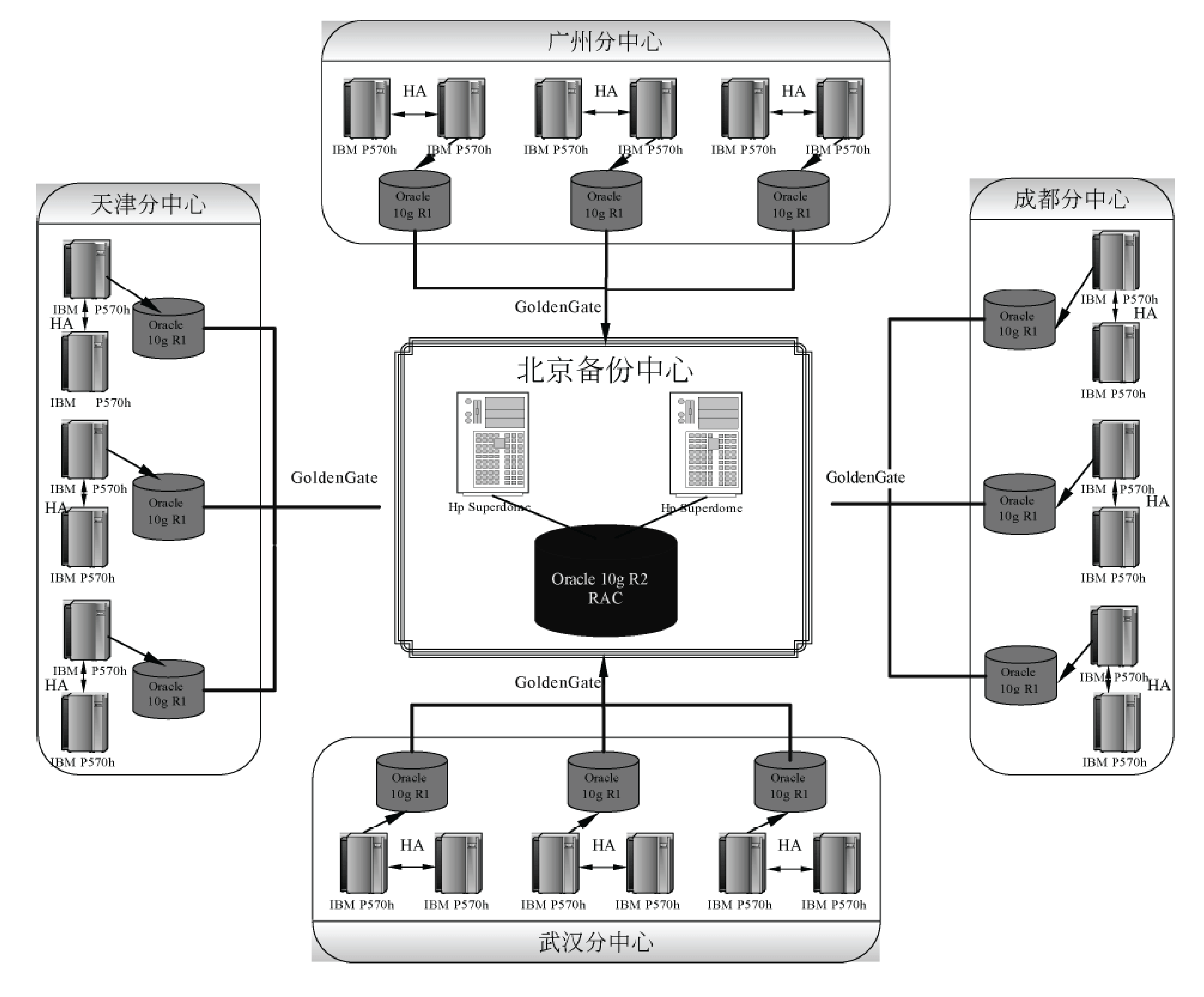


图 7-7

7.3.1　多源到单目标复制

多源到单目标复制是通过配置多个 Extract、Pump、Replicat 进程组最后把数据投递到一个库中，如图 7-8 所示。

多对一模式各链路相互独立，无法对 Replicat 进程合并，为了避免数据出现冲突，建议将多个源的数据存放于不同的实例、schema 或者表内。除非必选要把数据放在一个表中，否则强烈建议将每个 Replicat 进程的数据使用不同 schema 来存储。

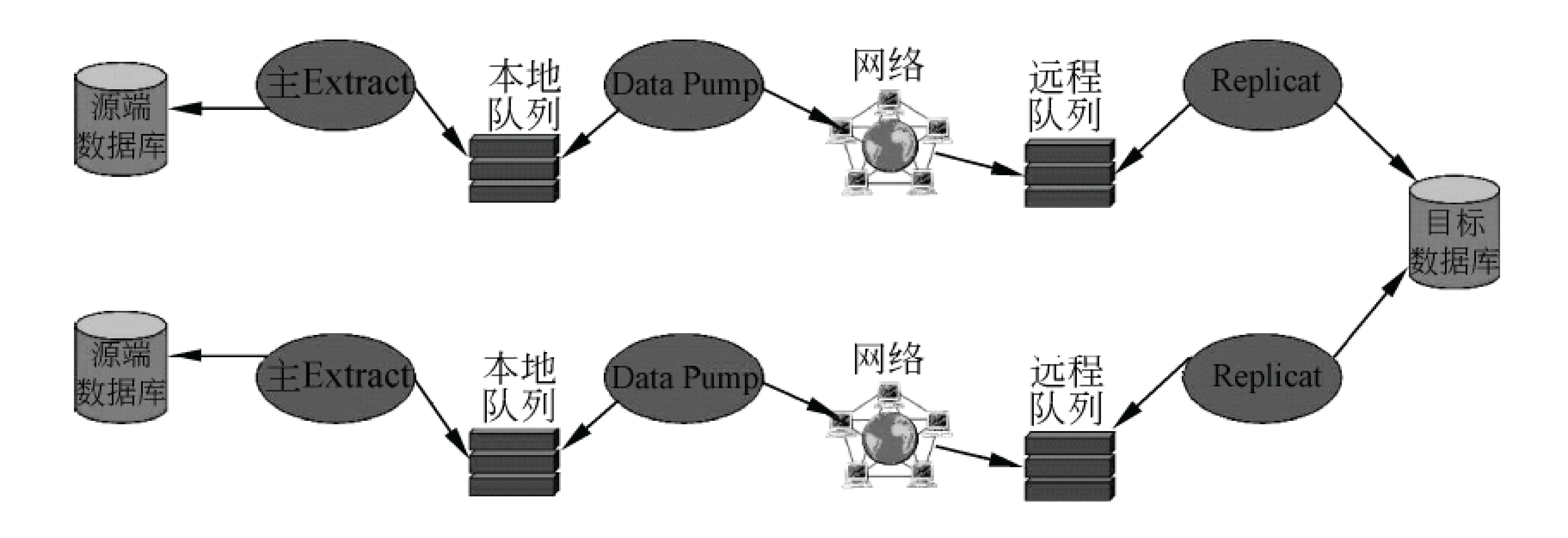


图 7-8

这里介绍二对一数据仓库的配置方法，与多对一配置方法类似。

（1）为每个系统创建一个 MGR 进程。

（2）源端使用 ADD EXTRACT 命令为每个系统添加主抽取进程组，分别叫做 ext_1 和 ext_2：

示例 7-18：

```
Extract 1
ADD EXTRACT <ext 1>, TRANLOG, BEGIN <time> [, THREADS <n>]
Extract 2
ADD EXTRACT <ext_2>, TRANLOG, BEGIN <time> [, THREADS <n>]
```

（3）源端使用 ADD EXTTRAIL 命令为每个系统添加 trail 文件：

示例 7-19：

```
Extract 1
ADD EXTTRAIL <local_trail_1>, EXTRACT <ext_1>
Extract_2
ADD EXTTRAIL <local_trail_2>, EXTRACT <ext_2>
```

（4）使用 EDIT PARAMS 命令为每个 Extract 进程组编辑参数：

示例 7-20：

```
Extract 1
EXTRACT <ext 1>
[SOURCEDB <dsn 1>,][USERID <user>[, PASSWORD <pw>]]
EXTTRAIL <local trail 1>
TABLE <owner>.<table>;

Extract_2
EXTRACT <ext 2>
[SOURCEDB <dsn 2>,][USERID <user>[, PASSWORD <pw>]]
EXTTRAIL <local trail 2>
TABLE <owner>.<table>;
```

（5）源端为每个系统添加 Pump 进程，分别为 pump_1 和 pump_2，要 EXTTRAILSOURCE 属性来标识需要 Pump 进程读取的 trail 文件：

示例 7-21：

```
Data pump_1
ADD EXTRACT <pump 1>, EXTTRAILSOURCE <local trail 1>, BEGIN <time>
Data pump 2
ADD EXTRACT <pump_2>, EXTTRAILSOURCE <local_trail_2>, BEGIN <time>
```

（6）源端使用命令 ADD RMTTRAIL 为 Pump 进程指定目标段 trail 文件：

示例 7-22：

```
remote 1
ADD RMTTRAIL <remote trail 1>, EXTRACT <pump 1>
remote_2
ADD RMTTRAIL <remote_trail_2>, EXTRACT <pump_2>
```

（7）为 Pump 进程添加参数：

示例 7-23：

```
Data pump_1
EXTRACT <pump 1>
[SOURCEDB <dsn 1>,][USERID <user>[, PASSWORD <pw>]]
RMTHOST <target>, MGRPORT <portnumber>
RMTTRAIL <remote trail 1>
[PASSTHRU | NOPASSTHRU]
TABLE <owner>.<table>;

Data pump 2
EXTRACT <pump 2>
[SOURCEDB <dsn 2>,][USERID <user>[, PASSWORD <pw>]]
RMTHOST <target>, MGRPORT <portnumber>
RMTTRAIL <remote trail 2>
[PASSTHRU | NOPASSTHRU]
TABLE <owner>.<table>;
```

（8）容灾端使用命令 ADD REPLICAT 添加 Replicat 进程组：

示例 7-24：

```
Replicat_1
ADD REPLICAT <rep_1>, EXTTRAIL <remote_trail_1>, BEGIN <time>
Replicat 2
ADD REPLICAT <rep_2>, EXTTRAIL <remote_trail_2>, BEGIN <time>
```

（9）为容灾端 Replicat 进程添加参数：

示例 7-25：

```
Replicat 1
REPLICAT <rep_1>
```

```
SOURCEDEFS <full pathname> | ASSUMETARGETDEFS
[TARGETDB <dsn 3>,] [USERID <user id>[, PASSWORD <pw>]]
REPERROR (<error>, <response>)
MAP <owner>.<table>, TARGET <owner>.<table>[, DEF <template name>];
Replicat_2
REPLICAT <rep 2>
SOURCEDEFS <full pathname> | ASSUMETARGETDEFS
[TARGETDB <dsn 3>,] [USERID <user id>[, PASSWORD <pw>]]
REPERROR (<error>, <response>)
MAP <owner>.<table>, TARGET <owner>.<table>[, DEF <template name>];
```

7.3.2 多表到单表复制

多表对单表复制，通常是多个数据结构相同或相近的数据汇总。

多表到单表的复制容灾经常出现数据存在相同主键或相同数据的问题，当容灾端出现重复记录的时候，可以使用参数 OVERRIDEDUPS 或 NOOVERRIDEDUPS 来控制 Replicat 进程是否覆盖目标库中已经存在的数据，通过参数 GETDELETES | IGNOREDELETES 来控制 GoldenGate 的 Replicat 是否忽略 delete 操作。

下面配置把 3 个表的数据复制到一个表中，是用 OVERRIDEDUPS 参数覆盖已经存在的记录，使用 IGNOREDELETES 参数让 Replicat 进程忽略删除操作。

示例 7-26：

```
OVERRIDEDUPSP
MAP ELISDATA.CLIENT BASE, TARGET LIFEDATA.client info;
IGNOREDELETES
MAP ELISDATA.CLIENT EXTEND, TARGET LIFEDATA.client info, &
colmap (usedefaults,
       PHONETICIZE LASTNAME = LASTNAME,
       PHONETICIZE_FIRSTNAME = FIRSTNAME
);
MAP ELISDATA.SITE EMAIL, TARGET LIFEDATA.client info,  &
colmap (usedefaults,
       E MAIL=EMAIL
);
```

7.4 级联复制

GoldenGate 支持级联复制同步，GoldenGate 从源端捕获数据到中间库，然后再从中间库到目标端数据库。

在下列 3 种情况下可以使用级联复制。

- ❑ 目标库和生产库没有直接的网络连接，而中间库可以同时连接到生产端和目标端。
- ❑ 用户想隔离目标库到生产库的直接网络连接。
- ❑ 生产库和目标在地理位置上相距非常的远。例如把数据从芝加哥到洛杉矶，需要在中国中转数据。

级联复制的原理图如图 7-9 所示。

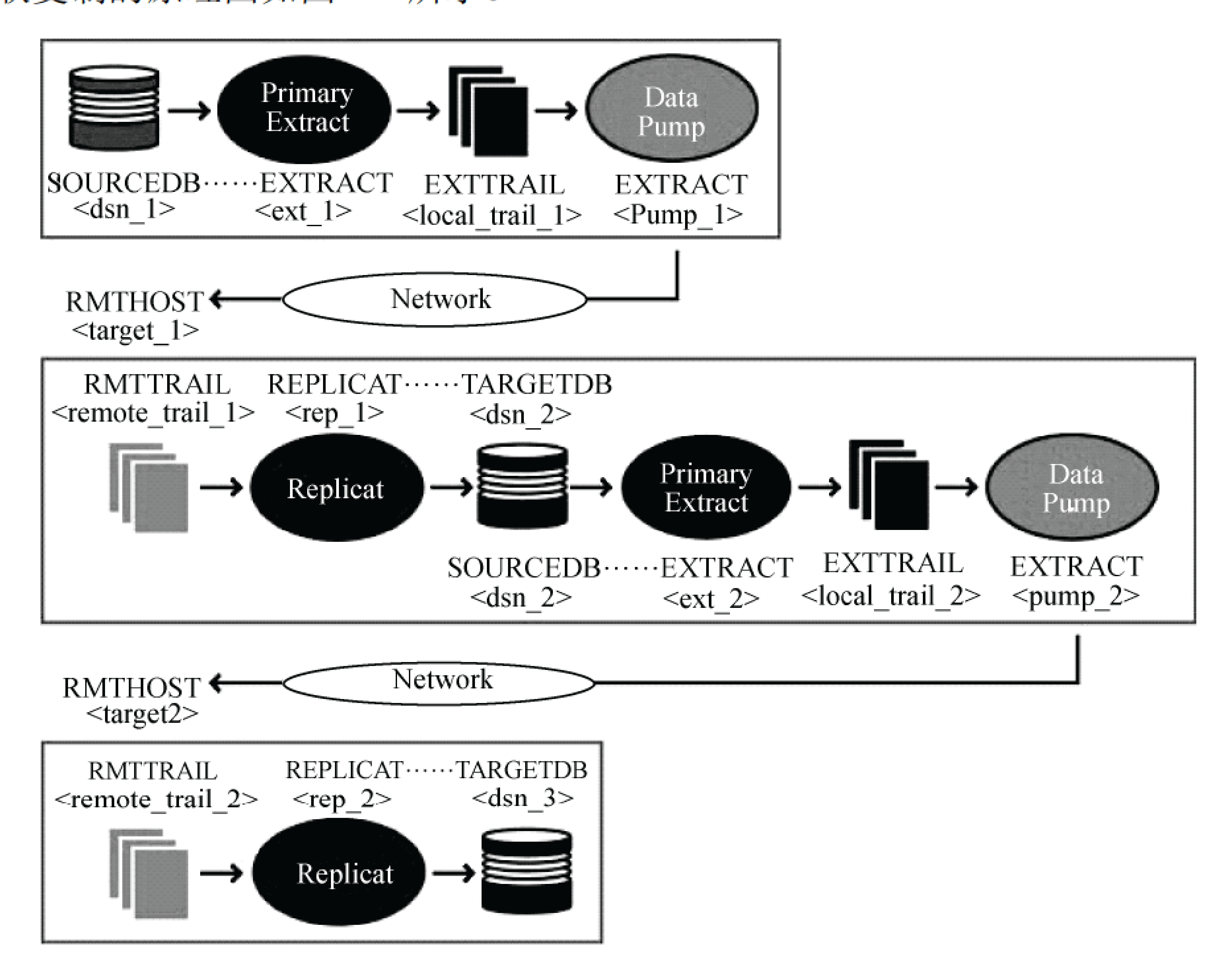


图 7-9

级联复制的原理：

- ❑ 生产端 Extract 进程组抽取数据变化到本地 trail 文件，Pump 进程传送数据到中间库。
- ❑ 中间库 Replicat 进程复制数据到中间库。
- ❑ 中间配置 Extract 进程抽取数据变化到中间库本地 trail 文件，中间库配置 Pump 进程组投递数据到目标库。
- ❑ 目标库 Replicat 进程复制数据到目标库。

级联复制是其他方式的组合，可以根据客户需求任意排列，级联复制的参数无特殊配置。级联复制可用于数据的逐级集中或者下发，县级数据备份到省市级，省市级备份到中央。

级联复制的配置方法为生产库配置、中间库配置、目标库配置。

7.4.1　生产库配置

（1）生产库使用 ADD EXTRACT 命令创建名字为 ext_1 的 Extract 进程组：

示例 7-27：

```
ADD EXTRACT <ext_1>, TRANLOG, BEGIN <time> [, THREADS <n>]
```

（2）生产库使用 ADD EXTTRAIL 命令创建本地 trail 文件：

示例 7-28：

```
ADD EXTTRAIL <local_trail_1>, EXTRACT <ext>
```

（3）使用 EDIT PARAMS 命令 ext_1 编辑参数：

示例 7-29：

```
EXTRACT <ext 1>
[SOURCEDB <dsn 1>,][USERID <user>[, PASSWORD <pw>]]
EXTTRAIL <local trail 1>
TABLE <owner>.<table>;
```

（4）在生产库使用 ADD EXTRACT 命令添加 Pump 进程组：

示例 7-30：

```
ADD EXTRACT <pump_1>, EXTTRAILSOURCE <local_trail_1>, BEGIN <time>
```

（5）在生产库使用 ADD REMOTETRAIL 命令为 Pump 添加远端 trail 文件：

示例 7-31：

```
ADD RMTTRAIL <remote_trail_1>, EXTRACT <pump_1>
```

（6）为生产库 Pump 进程添加参数：

示例 7-32：

```
EXTRACT <pump 1>
[SOURCEDB <dsn_1>,][USERID <user>[, PASSWORD <pw>]]
RMTHOST <target_1>, MGRPORT <portnumber>
RMTTRAIL <remote trail 1>
[PASSTHRU | NOPASSTHRU]
TABLE <owner>.<table>;
```

7.4.2 中间库配置

（1）中间库添加检查点表：

示例 7-33：

```
GGSCI (OE5) 5>edit params ./GLOBALS
Edit params ./GLOBALS
checkpointtable GoldenGate.ckpt

dblogin userid GoldenGate, password GoldenGate
```

```
add checkpointtable GoldenGate.ckpt
```

（2）中间库使用 ADD REPLICAT 命令添加 Replicat 进程：

示例 7-34：

```
ADD REPLICAT <rep_1>, EXTTRAIL <remote_trail_1>, BEGIN <time>
```

（3）中间库为 Replicat 进程编辑参数：

示例 7-35：

```
REPLICAT <rep 1>
SOURCEDEFS <full_pathname> | ASSUMETARGETDEFS
[TARGETDB <dsn_2>,] [USERID <user id>[, PASSWORD <pw>]]
REPERROR (<error>, <response>)
MAP <owner>.<table>, TARGET <owner>.<table> [, DEF <template name>];
```

（4）中间库添加名字为 ext_2 的 Extract 进程组：

示例 7-36：

```
ADD EXTRACT <ext_2>, TRANLOG, BEGIN <time> [, THREADS <n>]
```

（5）中间库为 ext_2 添加本地 trail 文件：

示例 7-37：

```
ADD EXTTRAIL <local_trail_2>, EXTRACT <ext_2>
```

（6）中间库为 ext_2 编辑参数文件：

示例 7-38：

```
EXTRACT <ext 2>
[SOURCEDB <dsn 2>,][USERID <user>[, PASSWORD <pw>]]
EXTTRAIL <local trail 2>
IGNOREAPPLOPS, GETREPLICATES
TABLE <owner>.<table>;
```

IGNOREAPPLOPS 和 GETREPLICATES 参数表示忽略中间库应用对数据的修改，中间库 Extract 进程只抽取 Replicat 进程产生的修改。

（7）中间库添加 Pump 进程，投递数据到目标端：

示例 7-39：

```
ADD EXTRACT <pump_2>, EXTTRAILSOURCE <local_trail_2>, BEGIN <time>
```

（8）中间库为 Pump_2 添加源端 trail 文件：

示例 7-40：

```
ADD RMTTRAIL <remote_trail_2>, EXTRACT <pump_2>
```

（9）中间库为 Pump_2 编辑参数文件：

示例 7-41：

```
EXTRACT <pump_2>
[SOURCEDB <dsn 2>,][USERID <user>[, PASSWORD <pw>]]
RMTHOST <target 2>, MGRPORT <portnumber>
RMTTRAIL <remote trail 2>
[PASSTHRU | NOPASSTHRU]
TABLE <owner>.<table>;
```

7.4.3　目标库配置

（1）目标库添加 Replicat 进程，把中间库抽取过来的数据复制到目标端数据库：

示例 7-42：

```
ADD REPLICAT <rep_2>, EXTTRAIL <remote_trail_2>, BEGIN <time>
```

（2）为目标库 rep_2 配置参数文件：

示例 7-43：

```
REPLICAT <rep 2>
SOURCEDEFS <full pathname> | ASSUMETARGETDEFS
[TARGETDB <dsn 3>,] [USERID <user id>[, PASSWORD <pw>]]
REPERROR (<error>, <response>)
MAP <owner>.<table>, TARGET <owner>.<table>[, DEF <template name>];
```

7.5　数据的转换

GoldenGate 对所有数据的选择、映射和过滤都是通过两个完整参数 table 和 map 来完成的。TABLE 用在 Extract 参数文件中，而 MAP 用在 Replicat 参数文件中。而在 TABLE 和 MAP 参数的基础上再添加一些属性，如 where、filter 等就可以实现数据的选择和过滤。

数据的过滤和转换发生在生产端、容灾端和中间数据库。但是不建议在 Pump 进程中实现数据的过滤和转换。为了减轻生产库的负担，建议在容灾端实现数据的转换。

7.5.1　数据选择与过滤

在生产端或容灾端使用 where 或 filter 可以实现数据的选择和过滤。由于 filter 比 where 能使用更多的函数，而 where 选择只是基本 SQL 的 where 操作，所以实现数据的选择和过滤，filter 比 where 使用更多。

实现转换的语法：

示例 7-44：

```
TABLE <table spec>,
```

```
, FILTER (
[, ON INSERT | ON UPDATE| ON DELETE]
[, IGNORE INSERT | IGNORE UPDATE | IGNORE DELETE]
, <filter clause>);

MAP <table spec>, TARGET <table spec>,
, FILTER (
[, ON INSERT | ON UPDATE| ON DELETE]
[, IGNORE INSERT | IGNORE UPDATE | IGNORE DELETE]
[, RAISEERROR <error num>]
, <filter clause>);
```

Filter 是基于数值的操作，在下列几种情况建议使用 filter 操作：

- 数值类型。
- 所操作的字段包括数值。
- 函数的返回值是数值类型。
- 所支持的数学操作的数据类型：+、-、*、/、\。
- 操作包含比较运算符：>、>=、<、<=、<>、=。
- 操作包含逻辑运算符：and or。

当使用 filter 属性时，下列的参数可以联合 filter 一起使用：

示例 7-45：

```
ON INSERT | ON UPDATE | ON DELETE
IGNORE INSERT | IGNORE UPDATE | IGNORE DELETE
```

当 filter 发生错误的时候在容灾端使用 RAISEERROR 参数可以唤起用户自定义错误信息。

在 filter 中使用字符串时必须加上双引号。

使用 filter 选择的一些典型的例子：

例一：调用 GoldenGate 的@COMPUTE 函数抽取价格操作 10000 的记录。

示例 7-46：

```
MAP sales.tcustord, TARGET sales.tord,
FILTER (@COMPUTE (product_price*product_amount) > 10000);
```

例二：调用 GoldenGate 的@STRFIND 函数抽取包含 joe 字符串的记录。

示例 7-47：

```
TABLE act.tcustord, FILTER (@STRFIND (name, "joe") > 0);
```

例三：当 select 记录大于 50 条时，在 UPDATE 和 DELETE 上执行 filter。

示例 7-48：

```
TABLE act.tcustord, FILTER (ON UPDATE, ON DELETE, amount > 50);
```

例四：调用 GoldenGate 的@RANGE 函数把表拆分到两个进程中去，这张情况在实际生产环境用的情况比较多。

示例 7-49：

```
(Replicat group 1 parameter file)
MAP sales.acct, TARGET sales.acct, FILTER (@RANGE (1, 2, ID));
(Replicat group 2 parameter file)
MAP sales.acct, TARGET sales.acct, FILTER (@RANGE (2, 2, ID));
```

使用 where 过滤的情况比较少，一般用在通过字符比较的操作中。

使用 where 选择数据的格式：

示例 7-50：

```
TABLE <table spec>, WHERE (<WHERE clause>);
Or…
MAP <table spec>, TARGET <table spec>, WHERE (<WHERE clause>);
```

Extract 端进行过滤，主要用于数据的分发、过滤敏感数据，对下游数据库进行有选择的数据传递。

- **优势** 控制数据安全，有选择的分发敏感数据、占用更少的磁盘空间。
- **劣势** 过滤数据使 Extract 日志能力有所下降。

Replicat 端进行过滤，主要用于过滤掉无用的数据。

- **优势** 只复制部分数据，可以间接实现数据的拆分，在读写分离的系统中也可以实现负载均衡。
- **劣势** 只复制部分数据，一样占用全部复制的带宽，如果在带宽有限的情况下，最好在 Extract 端进行过滤。

默认情况下 GoldenGate 选择和转换 INSERT、UPDATE 和 DELETE 操作。你可以在 Extract 和 Replicat 中使用下列参数来控制 GoldenGate 是否忽略某些操作。

示例 7-51：

```
GETINSERTS | IGNOREINSERTS          --是否忽略 insert 操作
GETUPDATES | IGNOREUPDATES          --是否忽略 update 操作
GETDELETES | IGNOREDELETES          --是否忽略 delete 操作
```

7.5.2 列映射

GoldenGate 支持 table 级别和全局级别的列映射。

1. table-level 列映射

下列两种需要在 TABLE 和 MAP 中使用 COLMAP 参数来进行列映射。

- 生产库的列和目标库的列的名字不同。
- 指定默认的列映射当明确的列映射不需要的时候。

COLMAP 参数提供从生产库的列到容灾库的列的选择、映射、转换和数据移动，在 COLMAP 参数中还可以包含 GoldenGate 的列转换函数来实现列的映射。

当生产端和容灾的数据结构不一致时，使用 COLMAP 参数还需要定义转换文件。

生产端 TABLE 容灾端 MAP，列映射 COLMAP 的语法：

示例 7-52：

```
TABLE <table spec>, TARGET <table spec>, &
COLMAP ([USEDEFAULTS, ] <target column> = <source expression>);
Or…
MAP <table spec>, TARGET <table spec>, &
COLMAP ([USEDEFAULTS, ] <target column> = <source expression>);
```

对这个语法的各个参数的解释：

TARGET <table spec>：目标的用户下的表名，前面会有 MAP。

<target column>：要复制到目标端的列名。

<source expression>：可以是要映射数据的以下的描述。

- 连续不断的数字，例如：123。
- 在双引号中包含的连续不断的字母，例如 ABCD。
- 生产端的列名，例如 ORD_DATE。
- GoldenGate 转换函数的表现，例如@STREXT (COL1, 1, 3)。

USEDEFAULTS：如果生产端和容灾端的列的名字一样，怎么使用默认的映射规则。

下面是一个同时用到默认映射和列一一对应映射的生产端表 ACCTBL 到容灾端表 ACCTTAB 使用列映射的典型案例，其中大多数的列相同，只有下列几种不同的情况。

- 生产端表有一个 CUST_NAME 列，而容灾端表有一个 NAME 列。
- 生产端表有一个十进制数 PHONE_NO 列需要把它分配到容灾端表 AREA_CODE、PHONE_PREFIX、PHONE_NUMBER 3 个列中。
- 生产端表的 3 列 YY、MM、DD 复制到容灾端的一个 TRANSACTION_DATE 列中。

看到上面的情况，使用 USEDEFAULTS 参数来自动匹配相同的列，而一一对应的映射和函数用于不同的列。下面是这个例子的参数配置。

示例 7-53：

```
MAP sales.acctbl,TARGET sales.accttab, --生产端到目标端表的映射
COLMAP (                                --开始 COLMAP 声明
USEDEFAULTS,                            --生产端和目标端表相同的列使用默认复制
name = cust_name,                       --生产端到目标端列的映射
transaction_date =                      --使用@DATE 函数合并生产端的列到目标端
@DATE ("YYYY-MM-DD", "YY",YEAR, 'MM", MONTH, "DD", DAY),
area_code =                   --使用@STREXT 函数把 phone_no 分配到容灾端的 3 列
@STREXT (phone_no, 1, 3),
phone_prefix =
@STREXT (phone_no, 4, 6),
phone_number =
```

```
@STREXT (phone_no, 7, 10));
```

2. 全局列映射

使用 COLMATCH 参数创建全局的列映射，使用 COLMATCH 参数可以映射具有相同的表结构和数据集，只有列名不同的两个表。COLMATCH 提供了比 table_level 只能在 TABLE 和 MAP 使用 COLMAP 更多的映射方法。

全局列映射的语法。

示例 7-54：

```
COLMATCH
{NAMES <target column> = <source column> |      --基于列名的映射
PREFIX <prefix> |                               --忽略前缀
SUFFIX <suffix> |                               --忽略后缀
RESET}                                          --后继的映射使用默认方式
```

表 7-1 是一个使用 COLMATCH 使用的例子，只有列名不同。

表 7-1

Source Database		**Target Database**	
ACCT Table	**ORD Table**	**ACCOUNT Table**	**ORDER Table**
CUST_CODE	CUST_CODE	CUSTOMER_CODE	CUSTOMER_CODE
CUST_NAME	CUST_NAME	CUSTOMER_NAME	CUSTOMER_NAME
CUST_ADDR	ORDER_ID	CUSTOMER_ADDRESS	ORDER_ID
PHONE	ORDER_AMT	PHONE	ORDER_AMT
S_REP	S_REP	REP	REP
S_REPCODE	S_REPCODE	REPCODE	REPCODE

生产端和目标端映射的参数与其他表的处理方式一样：

示例 7-55：

```
COLMATCH NAMES customer code = cust code
COLMATCH NAMES customer name = cust name
COLMATCH NAMES customer address = cust addr
COLMATCH PREFIX S
MAP sales.acct, TARGET sales.account, COLMAP (USEDEFAULTS);
MAP sales.ord, TARGET sales.order, COLMAP (USEDEFAULTS);
COLMATCH RESET
MAP sales.reg, TARGET sales.reg;
MAP sales.price, TARGET sales.price;
```

对这个例子的解释：

- ❑ 映射源端表 acct、ord 的 cust_code 列到目标表 account、order 的 customer_code 列。
- ❑ 忽略 S_前缀。
- ❑ 源和目标表相同的列自动使用 USEDEFAULTS 默认映射。

- ❑ 关闭刚才的映射，表 sales.reg 和 sales.price 使用默认映射方式。

3．映射数据类型

下面介绍 GoldenGate 如何映射数据类型。

数据类型的列：数据类型映射目标列的类型和度量，如果目标的大小小于源端的列，数据将被截去右面的部分，如果目标的大小大于源端的类，则右面补零。

包含文字和数字的列：字符类型的列可以是 varchar、group、date、time 或者包含在双引号中的字符串类型。如果目标端列宽小于生产端，截去右面的多余的部分，如果目标的列宽大于生产端的列，多余的部分补为空格。

时间日期的列：时间日期包含 date、time 或 timesatamp 类型的列。映射时间日期类型的类，要确保遵守 GoldenGate 的外部格式 YYYY-MM-DD:HH:MI:SS.FFFFFF，如果目标端列小于源端，则截去多余的部分，如果目标端列大于源端的部分，则继续补上当前更详细的时间和日期。

7.5.3　函数功能

使用 GoldenGate 的函数功能可以把源端合适的数据投递到目标端列，GoldenGate 的函数可以使你操作数字和字符、执行测试、提取参数值、返回系统环境变量等。

在数据选择和过滤、列映射的章节已经使用了 GoldenGate 的很多函数，相信读者已经对函数的使用方法有了一定的了解。

使用 GoldenGate 函数的语法：

示例 7-56：

```
@<function name> (<argument>)
```

对语法词汇的解释：

@<function name>：GoldenGate 的函数名称，加上前缀“@”符号，例@DATE。

<argument>：函数的作用的数据，可以包含下列类型。

- ❑ 连续不断的数字，例如 123。
- ❑ 双引号内连续不断的字符，例如 ABCD。
- ❑ 源端的列名，例如 PHONE_NO。
- ❑ 逻辑运算，例如 COL2 * 100。
- ❑ 比较字符串，例如 COL3 > 100 AND COL4 > 0。
- ❑ 其他 GoldenGate 函数，例如 AMOUNT = @IF (@COLTEST (AMT, MISSING, INVALID), 0, AMT)。

1．函数的使用案例

判断数据：

示例 7-57：

```
@IF (SR_AREA = 20, "TRUE", "FALSE")
```

转换日期：
示例 7-58：

```
ORDER FILLED = @DATE (
"YYYY-MM-DD:HH:MI:SS",
"JTS",
@DATE ("JTS",
"YYMMDDHHMISS",
ORDER TAKEN TIME) +
ORDER_MINUTES * 60 * 1000000)
```

使用@COMPUTER 返回算数运算的结果：
示例 7-59：

```
@COMPUTE (COL1 > 0 AND COL2 < 3) 返回 0.
@COMPUTE (COL1 < 0 AND COL2 < 3) 返回 0. COL2 < 3 is never evaluated.
@COMPUTE ((COL1 + COL2)/5) 返回 7.
```

2. 使用@COMPUTE 函数

当数据列丢失、不可用或为空时，@COMPUTE 函数返回一致的结果：
例子：如果 BALANCE 是 1000，但是 AMOUNT 是 NULL，则函数返回 NULL。
示例 7-60：

```
NEW_BALANCE = @COMPUTE (BALANCE + AMOUNT)
```

3. 使用@COLSTAT 函数

例一：使用@COLSTAT 返回指定的值。
示例 7-61：

```
ITEM = @COLSTAT (NULL)
```

这个将在目标段 ITEM 列返回 NULL。
例二：假如价格和质量为零，则返回零。
示例 7-62：

```
ORDER_TOTAL = @IF (PRICE < 0 AND QUANTITY < 0, PRICE *QUANTITY,
@COLSTAT(NULL))
```

4. 使用@COLTEST 函数

使用@COLTEST 函数来检查下列的情景。
- PRESENT 检查列是否为可显和不为空。
- NULL 检查列是否为可显和不为空。
- MISSING 检查列是否为不可显示的。

- INVALID 检查列是否为不可用的。

示例 7-63：

```
@COLTEST (AMOUNT, NULL, INVALID)
```

5．使用@IF 函数

使用@IF 函数在特定情况下返回两个值中的一个：

示例 7-64：

```
NEW BALANCE = @IF (@COLTEST (BALANCE, NULL, INVALID) OR @COLTEST (AMOUNT,
NULL, INVALID), @COLSTAT (NULL), BALANCE + AMOUNT)
```

这个函数返回下列值中的一个。

- **NULL**　当 BALANCE 是 NULL 或 INVALID。
- **MISSING**　但任何一个列不能显示的时候。
- 这两个列的和。

6．使用@CASE 函数

使用@CASE 函数选择一个值。

示例 7-65：

```
@CASE (PRODUCT_CODE, "CAR", "A car", "TRUCK", "A truck")
```

这个函数返回下列值中的一个。

- **"A car"**　如果 PRODUCT_CODE 的值是"CAR"。
- **"A truck"**　如果 PRODUCE_CODE 的值是"TRUCK"。
- **"FIELD_MISSING"**　如果 PRODUCE_CODE 的值不是上面两种。

7．使用@EVAL 函数

示例 7-66：

```
@EVAL (AMOUNT > 10000, "high amount", AMOUNT > 5000, "somewhat high")
```

这个函数返回下列值中的一个。

- **"high amount"**　如果 AMOUNT 的值大于 10000。
- **"somewhat high"**　如果 AMOUNT 大于 5000 小于 10000。
- **"FIELD_MISSING"**　如果 PRODUCE_CODE 的值不是上面两种。

7.6　双业务中心场景

通常双业务中心分两种情况 Primary-Standby 的主备模式和 Active-Active 的双活模式。其中 Primary-Standby 模式通常只能在一个端进行增删改，那么这一端为 Primary，另一端供报表查询或者统计之用，称这一端为 Standby。Active-Active 则是两边的地位均等，两边

都可读写。Primary-Standby 模式适用于容灾和读写分离，Active-Active 主要适用于两边对等的业务中心。

7.6.1 Primary-Standby 模式切换

GoldenGate 支持将数据从一个活动的主库复制到一个备库中，为了防止计划的和非计划的停机事故。

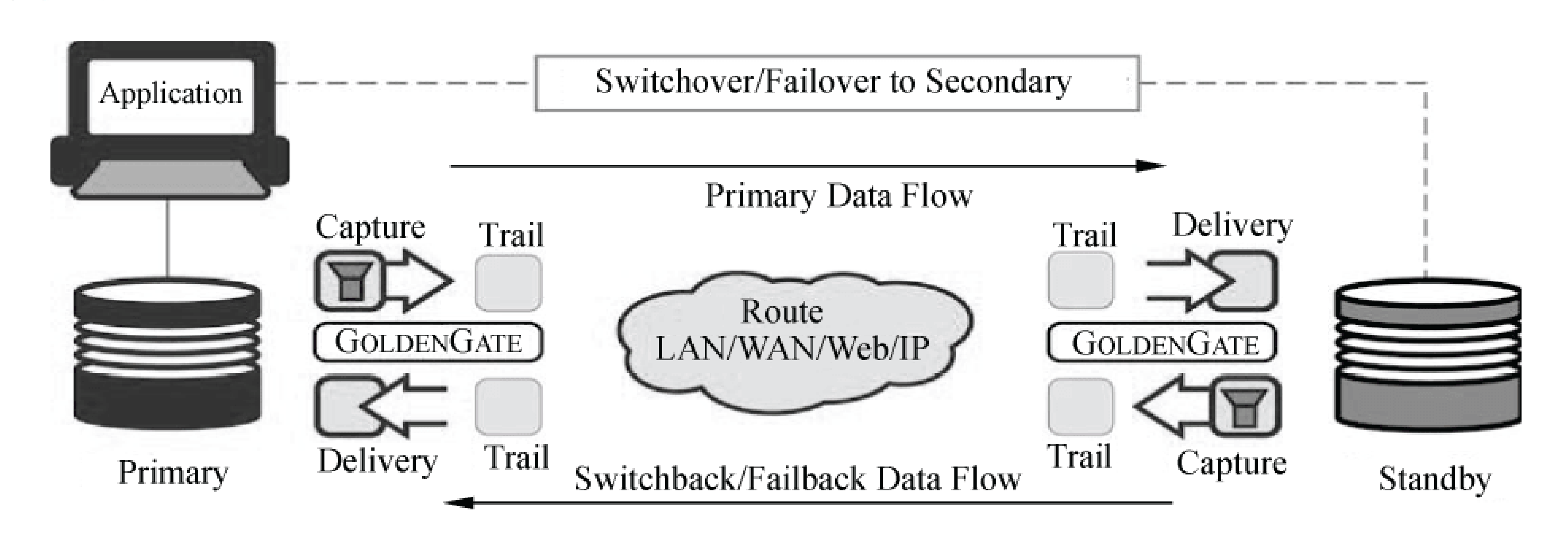


图 7-10

在这种配置中，虽然有两组 Extract-Pump-Replicat 配置，但是只有一组主库到备库的配置是活动的。只有当用户把业务从主库上移动到备库上的时候才启动备库到主库的进程组。当用户把应用移到备库上的时候，备库上的 Extract 进程将开始抽取备库上的数据变化，并把它放在本地的 trail 文件中，直到主库再次可用。

当主机恢复以后，GoldenGate 管理进程和 Replicat 进程立即开始运行，备库上存储的数据修改复制到生产库中，以实现两库的数据对等。在适当的时间，GoldenGate 可以把应用从备库移动到主库。GoldenGate 配置成正常的 Primary-Standby 模式，为以后可能出现的数据库 DOWN 机做准备。

1. 配置 Primary-Standby 需要考虑的问题

正常的业务操作都发生在主库，所有的备库都是获取主库的变化来使数据再同步，所以在配置 Primary-Standby 模式的时候需要备份主库然后在备库上恢复来实现初始化同步。

实施 Primary-Standby 模式的时候，主库和备库的容量和结构要尽量的一致。数据的选择、转换和过滤在这种数据库上不适用。

如果允许可以利用备库来实现报表和查询，但不可以是改变备库数据的 DML 操作。如果你想使备库上也可以有 DML 的业务操作，你可以配置 GoldenGate 为 Active-Active 模式。

在正常情况下，只有主库上的 Extract、Pump 进程和备库上的 Replicat 进程处在 running 状态。而备库上的 Extract、Pump 进程和主库的上 Replicat 要处在停止状态，以防在备库上发生的一个 DML 操作 GoldenGate 把它复制到主库上而影响主库的正常业务。

要定期的备份主库和备库上的 GoldenGate 的工作目录。必要的情况下备份 GoldenGate 安装目录的所有的文件和子目录。备份 GoldenGate 的工作目录，意味着在 GoldenGate 出

现失败的时候，用户不需要重建进程和参数文件。

在用户有计划的切换或不期望的停机的时候，要确保在主库和备库做好以下准备。

- 赋予用户 INSERT、UPDATE、DELETE 的脚本。
- 在备库上的启动 trigger 和删除级联约束的脚本。
- 切换应用服务器、启动应用服务器、copy 基本环境变量文件的脚本。
- 当主库 DOWN 机的时候把用户移动到备库的存储过程。

如果使用自动数据库自动生成的 KEY，变化的 KEY 可能在生产端和容灾端不一致。如果系统允许，用户可以添加自己的主键到数据库来保证主库和备库的一致。GoldenGate 通过使用 SQUENCE 和 MAP 参数来保证主库和备库的主键是一致的。

配置 Primary-Standby 的方法这里就不再赘述了。无非是配置两组 Extract-Pump-relicat 进程。一组是从主库和备库的，一组是备库到主库的。下面介绍一下主库到备库的切换方法。

2. 有计划迁移应用

这是一个使用 GoldenGate 同步数据，用户有计划的把应用从主库切换到备库上，然后在主库上执行一些维护操作，然后再切换的案例。

（1）把应用从主库移动到备库。

① 在主库上停止所有的用户应用程序，但是让 GoldenGate Extract 进程和 Pump 进程持续运行，来捕获可能的后台操作在数据库的修改。

② 在主库上输入以下命令，直到返回“At EOF, no more records to process”信息，表明所有的事务已经被捕获。

示例 7-67：

```
GGSCI (OE5) 2> lag extract ext 1

Sending GETLAG request to EXTRACT EXT_1 …
No records yet processed.
At EOF, no more records to process.
```

③ 在主库上，停止抽取进程：

示例 7-68：

```
GGSCI (OE5) 3> stop ext 1

Sending STOP request to EXTRACT EXT 1 …
Request processed.
```

④ 在主库的 Pump 进程上输入以下命令，直到返回“At EOF, no more records to process.”信息，表明 Pump 进程把所有的数据传送到了备库上：

示例 7-69：

```
GGSCI (OE5) 2> lag extract pump 1
```

```
Sending GETLAG request to EXTRACT pump_1 …
No records yet processed.
At EOF, no more records to process.
```

⑤ 在主库上，停止 Pump 进程：

示例 7-70：

```
GGSCI (OE5) 3> stop  pump_1

Sending STOP request to EXTRACT pump_1 …
Request processed.
```

⑥ 在备库上，输入以下命令，直到返回 “At EOF (endof file).” 信息，表明所有的数据已经复制到了备库上：

示例 7-71：

```
GGSCI (OE5) 10> lag rep_1

Sending GETLAG request to REPLICAT REP_1 …
Last record lag: 9 seconds.
At EOF, no more records to process.
```

⑦ 在备库上停止 Replicat 进程：

示例 7-72：

```
GGSCI (OE5) 3> stop  rep 1

Sending STOP request to REPLICAT rep 1 …
Request processed.
```

⑧ 在备库上做以下操作。

- 赋予应用用户对备库的 INSERT、UPDATE、DELETE 权限。
- 执行启用 trigger 和级联删除约束的脚本。
- 执行切换应用，启动应用和拷贝必须文件的脚本。

⑨ 在备库上，修改 Extract 进程从当前时间开始抽取。否则 Extract 会从使用 ADD EXTRACT 命令创建它的时间开始抽取。

示例 7-73：

```
GGSCI (OE5) 11> alter extract ext 2 , begin now
EXTRACT altered.
```

⑩ 在备库上启动 Extract 进程来准备抽取数据库中的事务变化：

示例 7-74：

```
GGSCI (OE5) 12> start ext 2

Sending START request to MANAGER …
```

```
EXTRACT EXTMA starting
```

⑪ 在备库上激活应用程序，让用户可用。

⑫ 在主库上执行维护操作。

（2）把应用从备库移回到主库

① 在备库上，停止所有的用户应用程序，但是让 Extract 进程运行，来获取可能后台进程对数据库的修改。

② 在主库上，启动 Replicat 进程准备来复制应用程序在备库上对数据的修改：

示例 7-75：

```
GGSCI (OE5) 12> start rep_2

Sending START request to MANAGER …
REPLICAT rep_2 starting
```

③ 在备库上，启动 Pump 进程把存储在备库上的队列文件通过 TCP/IP 协议传送到主库上：

示例 7-76：

```
GGSCI (OE5) 12> start pump_2

Sending START request to MANAGER …
EXTRACT pump_2 starting
```

④ 在备库上，输入下列命令，直到返回“At EOF, no more records to process.”信息，表明所有的数据已经抽取到了本地 trail 文件：

示例 7-77：

```
GGSCI (OE5) 2> lag extract ext 2

Sending GETLAG request to EXTRACT EXT 2 …
No records yet processed.
At EOF, no more records to process.
```

⑤ 在备库上，停止抽取进程：

示例 7-78：

```
GGSCI (OE5) 3> stop ext_2

Sending STOP request to EXTRACT EXT_2 …
Request processed.
```

⑥ 在备库上，执行下列命令，直到返回“At EOF, no more records to process.”信息，表明所有的队列文件已经传送到了主库上：

示例 7-79：

```
GGSCI (OE5) 2> lag extract pump_2
```

```
Sending GETLAG request to EXTRACT pump 2 …
No records yet processed.
At EOF, no more records to process.
```

⑦ 在备库上，停止 Pump 进程：

示例 7-80：

```
GGSCI (OE5) 3> stop  pump 2

Sending STOP request to EXTRACT pump_2 …
Request processed.
```

⑧ 在主库上，执行下列命令，直到返回“At EOF (end of file).”信息，表明所有的数据已经复制到了主库上：

示例 7-81：

```
GGSCI (OE5) 10> lag rep 2

Sending GETLAG request to REPLICAT REP 2 …
Last record lag: 9 seconds.
At EOF, no more records to process.
```

⑨ 在主库上，停止 Replicat 进程：

示例 7-82：

```
GGSCI (OE5) 3> stop  rep_2

Sending STOP request to REPLICAT rep_2 …
Request processed.
```

⑩ 在主库上做以下操作。

- 赋予应用用户对备库的 INSERT、UPDATE、DELETE 权限。
- 执行启用 trigger 和级联删除约束的脚本。
- 执行切换应用，启动应用和拷贝必须文件的脚本。

⑪ 在主库上，修改 Extract 进程从当前时间开始抽取。否则 Extract 会从使用 ADD EXTRACT 命令创建它的时间开始抽取。

示例 7-83：

```
GGSCI (OE5) 11> alter extract ext_1 , begin now
EXTRACT altered.
```

⑫ 在主库上，启动 Extract 进程来获取主库的数据变化：

示例 7-84：

```
GGSCI (OE5) 12> start ext_1
```

```
Sending START request to MANAGER …
EXTRACT EXTMA starting
```

⑬ 在主库上，激活用户应用程序。

⑭ 在主库上，启动 Pump 进程：

示例 7-85：

```
GGSCI (OE5) 12> start pump 1

Sending START request to MANAGER …
EXTRACT pump_1 starting
```

⑮ 在备库上，启动 Replicat 进程：

示例 7-86：

```
GGSCI (OE5) 12> start rep_1

Sending START request to MANAGER …
REPLICAT rep_1 starting
```

3. 数据库意外宕机切换

（1）从主库移动应用到备库

把主库应用迁移到备库做了以下操作。

- 确保备库可用。
- 确保主库的交易已经完全迁移到备库。
- 启动备库的 Extract 进行来抽取备库的数据变化。
- 用户应用程序迁移到备库。

在备库上做以下操作。

① 执行下列命令，直到返回“At EOF (end of file).”信息，表明所有的数据已经复制到了备库上了：

示例 7-87：

```
GGSCI (OE5) 41> lag rep_1

Sending GETLAG request to REPLICAT REPMA …
Last record lag: 120 seconds.
At EOF, no more records to process.
```

② 停止备库的 Replicat 进程：

示例 7-88：

```
GGSCI (OE5) 42> stop  rep 1
```

```
Sending STOP request to REPLICAT rep_1 …
Request processed.
```

③ 赋予应用用户对备库的 insert，update，delete 权限。

④ 执行启用 trigger 和级联删除约束的脚本。

⑤ 执行切换应用，启动应用和拷贝必须文件的脚本。

⑥ 启动备库的抽取进程来获取数据的变化：

示例 7-89：

```
GGSCI (OE5) 43> start ext 2

Sending START request to MANAGER …
EXTRACT EXTMA starting
```

⑦ 移动用户应用到备库，并让应用开始运行。

（2）把应用从备库移回来

从备库移回来做以下操作。

- 恢复 GoldenGate 的环境。
- 备份备库的数据并恢复到主库。
- 当备份开始的时候传送用户交易的变化。
- 统一备份的数据使其与备库的一致。
- 把用户应用移回到主库上。
- 准备继续维护 Primary-Standby 模式。

① 恢复主库上 GoldenGate 环境。

② 恢复主库上 GoldenGate 的目录。

③ 在主库上，运行 GGSCI 命令。

④ 在主库上，删除 Extract 进程：

示例 7-90：

```
GGSCI (OE5) 1> DELETE EXTRACT <ext 1>
EXTRACT deleted.
```

⑤ 在主库上，删除本地 trail 文件：

示例 7-91：

```
DELETE EXTTRAIL <local_trail_1>
```

⑥ 在主库上，重新添加抽取进程组，要和原来的抽取进程组使用同样的名字，为了和备份的 GoldenGate 目录中的参数文件中的名字匹配：

示例 7-92：

```
ADD EXTRACT <ext_1>, TRANLOG, BEGIN <time> [, THREADS <n>]
```

⑦ 为主抽取进程添加 trail 文件，为了匹配参数的文件的名字，起和原来相同的名字：

示例 7-93：

```
ADD EXTTRAIL <local_trail_1>, EXTRACT <ext_1>
```

⑧ 在主库上，启动管理进程。
⑨ 复制备库上的数据，到生产端。
⑩ 在主库上禁用 trigger 和 check 和外键约束。
⑪ 在备库上，开始热备份数据库。
⑫ 在备库上，记录备份完成的时间。
⑬ 在备库上，禁止用户访问数据库，并等待所有的事物完成。
⑭ 把复制过程的对数据库的操作应用到主库，确保数据一致。
⑮ 在主库上，启动 Replicat 进程组：
示例 7-94：

```
START REPLICAT <rep_2>
```

⑯ 在备库上，启动 Pump 进程组，把容灾端发生的事物变化应用到主库上：
示例 7-95：

```
START EXTRACT <pump_2>
```

⑰ 使用 info 命令，直到看到 Replicat 进程的时间已经追到了备份完成的时间，然后去掉 HANDDLECOLLSION 冲突处理操作：
示例 7-96：

```
SEND REPLICAT <rep_2>, NOHANDLECOLLISIONS
```

⑱ 在主库上是用 stats replicat 复制进程名命令，直到看到"At EOF (end offile)"信息，说明 GoldenGate 已经把所有的变化数据应用到了主库：
示例 7-97：

```
STATS rep_2
```

⑲ 停止主库上 Replicat 进程组，停止备库上的 Pump 进程：
示例 7-98：

```
STOP EXTRACT <pump 2>
STOP REPLICAT <rep_2>
```

⑳ 在主库上授予应用系统用户 INSERT、UPDATE、DELETE 操作的权限。
㉑ 在主库上启动 trigger 和外键约束。
㉒ 把应用迁移回主库上，然后启动主库上的 Extract 抽取进程。

7.6.2　配置双向复制 Active-Active 模式

GoldenGate 支持 Active-Active 模式的配置，两个系统都在发生数据变化，而且把每个

系统的数据变化应用到另一个库中，使两个库中的数据保持一致。

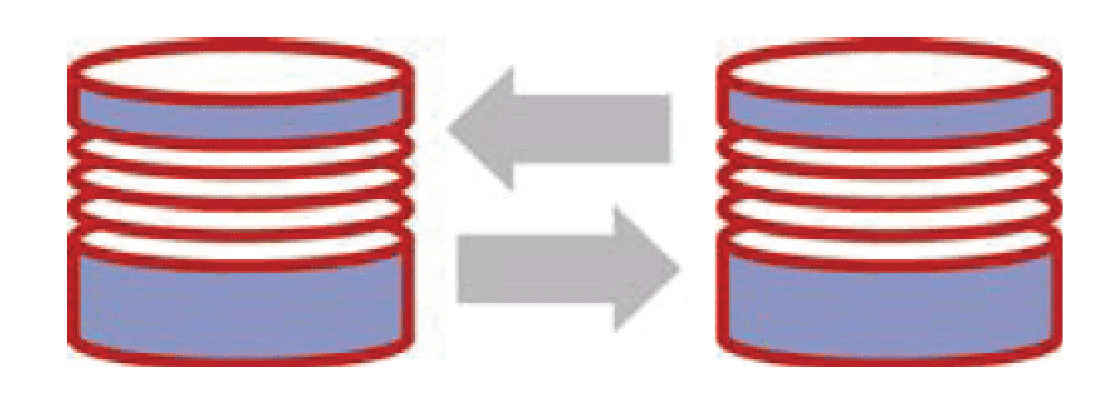


图 7-11

在 Active-Active 双向模式配置中，只有两组激活的 Extract-Pump-Replicat 进程组在同时工作，然后分别把各个系统的变化应用到另一个库中。在这种模式中 GoldenGate 不支持 DDL 操作。在这种配置中，所有的记录必须含有主键或唯一索引。Triggers and ON DELETE CASCADE 约束产生的 DML 操作会被 GoldenGate 抽取并应用。

为了防止本地的触发器和级联删除的操作和 GoldenGate 复制过来的触发器和级联删除操作冲突，需要做以下操作。

- ❑ 修改 trigger，忽略由 Replicat 进程复制过来的操作（trigger.cascade）。
- ❑ 禁用 cascade delete。通过在父表上创建 trigger 来实现级联删除。创建为 before trigger。这样在删除发生在父表之前，子表就先发生删除操作。

防止循环：在这种 Active-Active 模式中，很容易出现循环复制的情况。为了避免循环的出现，要做以下两种措施。

- ❑ 防止 Extract 进程抽取由 Replicat 进程复制过来的 DML 操作。
- ❑ 识别本地的 Replicat 事务，以便于抽取忽略它们。

GoldenGate 使用参数来忽略特定用户的操作，可以使用 userID 或 username。

示例 7-99：

```
TRANLOGOPTIONS EXCLUDEUSER <user name>
TRANLOGOPTIONS EXCLUDEUSERID <user-id>
```

配置 GoldenGate activate-activate 模式，如图 7-12 所示。

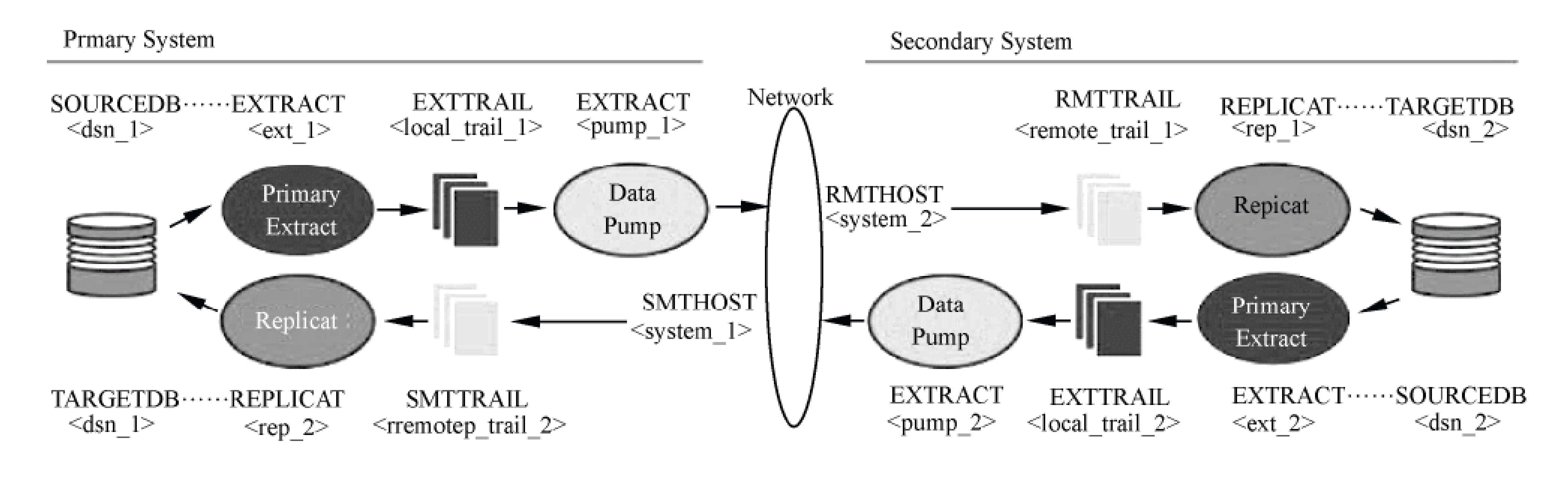


图 7-12

双向复制模式配置只需要配置两条复制链路，再加上一些冲突解决的方式（下面会介绍到），这里就不做配置的赘述了。

双向复制模式特点：

- 配置两个方向相反的复制链路。
- 使用 OGG 的冲突循环检测机制避免循环复制。
- 建议仅一端开展业务或者两端开展不同业务，如需两端同时开展业务需考虑同时对数据进行修改时的处理逻辑。
- 常用于双业务中心，是最理想的灾备模式。

双活的可行性：

- 双活只有通过逻辑数据复制技术（如 OGG）才能实现。
- 在应用程序和数据库配合下方能实现。

双中心的典型案例如图 7-13 所示。

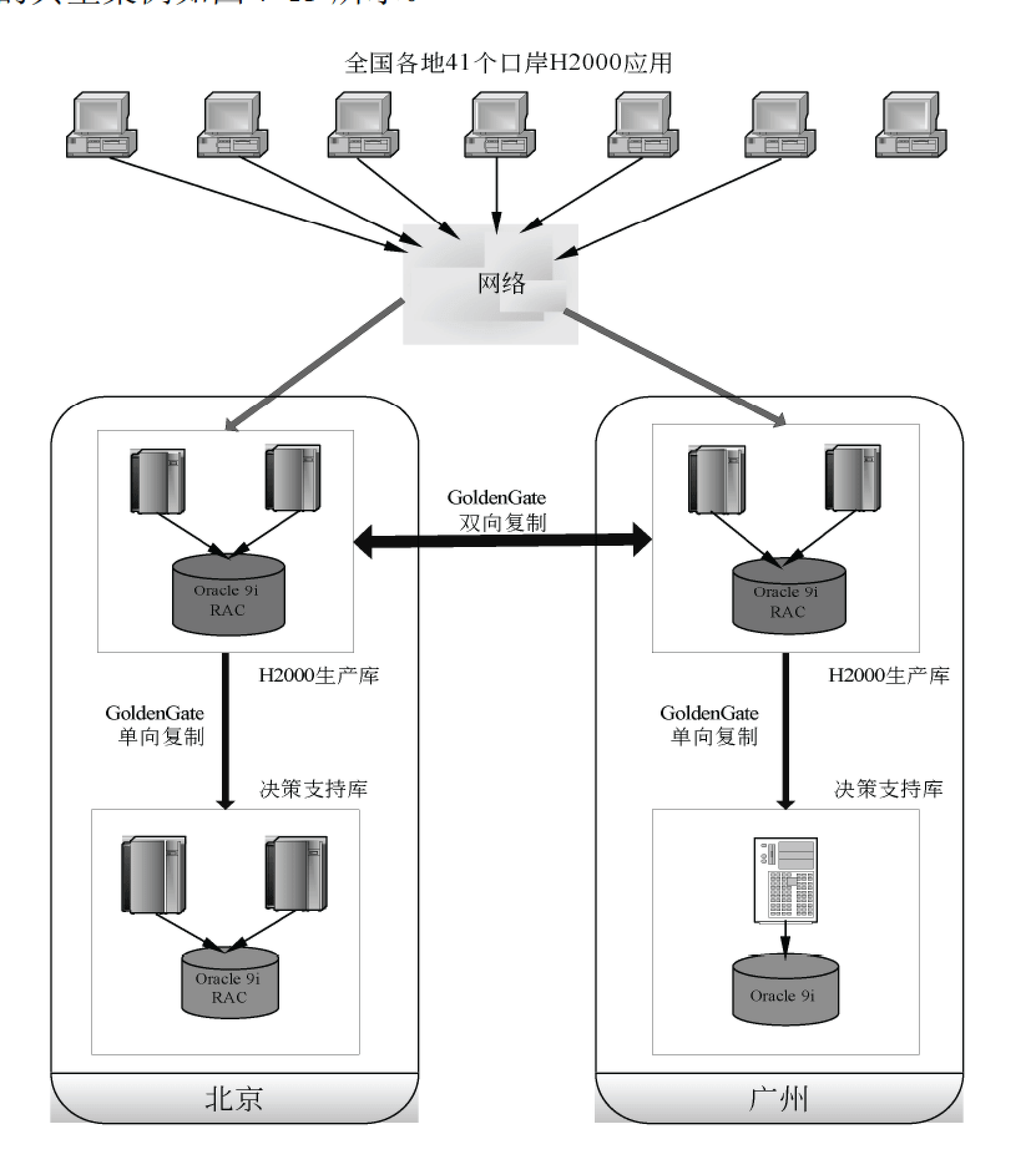


图 7-13

双活的核心是如何避免出现数据冲突。

- 第一选择：只在一端开展业务。
- 第二选择：两端开展不同业务，涉及不同数据集。
- 第三选择：两端开展相同业务，但依据地域或其他条件对数据予以区分，两边不操作同一条数据。

不建议两边对同一数据进行操作。如确实需要，则必须根据业务需要定义数据冲突处理机制，以下为几个推荐机制。

- ❑ 以时间戳为准，即比较记录的操作时间戳，只应用时间晚的数据。
- ❑ 比较前后差异，只应用两者之间的差值。
- ❑ 使用异常表记录数据冲突，由人工进行处理。

OGG 针对各种数据冲突处理的机制均需人工根据需求进行定制。

- ❑ OGG 不支持 DDL 双向复制。
- ❑ OGG 不支持 Sequence 双向复制。

7.6.3 Active-Active 冲突处理及解决

OGG 避免循环复制的原理：

- ❑ 在 Replicat 写入数据时做标记。
- ❑ 通过记录执行该交易的用户名或者 ID。
- ❑ 通过在交易中加入一条对 Trace table 操作的记录。
- ❑ 在 Extract 时根据 Replicat 所做记录忽略对应交易。

GolenGate 的相关参数。

- ❑ Extract 中缺省的 GETAPPLOPS 和 IGNOREREPLICATES，控制只读取应用程序交易、忽略 Replicat 写入的交易。
- ❑ Extract 中指定排除特定用户交易的参数：

示例 7-100：

```
TRANLOGOPTIONS EXCLUDEUSER: DB2/Ingres/Sybase/Oracle 10g+
TRANLOGOPTIONS EXCLUDETRANS: MS SQL Server/Sybase
```

- ❑ 配置实用 Trace Table 的参数（仅 Oracle 9i）：

示例 7-101：

```
GGSCI>ADD TRACETABLE [<owner>.<table name>]
```

注：请先 dblogin 登录数据库，在对应的 Ext/Rep 参数中添加：

示例 7-102：

```
TRACETABLE [<owner>.<table name>]
```

第 8 章　GoldenGate 日常维护

当 GoldenGate 软件部署实施完毕之后，接下来一部分非常重要的工作就是 GoldenGate 的日常维护与异常处理；这也是非常体现技术含量与一个工程师睿智的环节。

下面就来介绍这一部分内容，希望能对工作尽可能的有所帮助。

日常维护包括监视进程的运行状态，调优 GoldenGate 各进程处理性能与排除日常故障。做到这些的话经常需要利用以下工具获得足够多的诊断信息。

- GGSCI 命令行。
- Oracle GoldenGate 的跟踪参数。
- Oracle GoldenGate 的 report 文件与日志。
- 操作系统的日志。
- 用 logdump 工具分析 trail 文件，等等。

8.1　长事务处理

（1）检查长交易的存在。

在停止抽取进程前需要通过命令检查是否存在长交易，以防止下次启动无法找到归档日志：

示例 8-1：

```
GGSCI> info extXX, showch         （注：查看抽取进程的检查点）
Read Checkpoint #1
Recovery Checkpoint (position of oldest unprocessed transaction in the data
source):
    Thread #: 1
    Sequence #: 9671
    RBA: 239077904
    Timestamp: 2008-05-20 11:39:07.000000
    SCN: 2195.1048654191
    Redo File: Not available

Current Checkpoint (position of last record read in the data source):
    Thread #: 1
    Sequence #: 9671
    RBA: 239377476
    Timestamp: 201-02-20 11:39:10.000000
    SCN: 2195.1048654339
    Redo File: Not Available
```

```
Read Checkpoint #2
…

Recovery Checkpoint (position of oldest unprocessed transaction in the data
source):
    Thread #: 2
    Sequence #: 5287
    RBA: 131154160
    Timestamp: 2011-02-20 11:37:42.000000
    SCN: 2195.1048640151
    Redo File: /dev/rredo07

Current Checkpoint (position of last record read in the data source):
    Thread #: 2
    Sequence #: 5287
    RBA: 138594492
    Timestamp: 2011-02-20 11:39:14.000000
    SCN: 2195.1048654739
    Redo File: /dev/rredo07
```

为了方便长交易的管理，GoldenGate 提供了一些命令来查看这些长交易，可以帮助客户和应用开发商查找到对应的长交易，并在 GoldenGate 中予以提交或者回滚。

查看长交易的方法：

示例 8-2：

```
Ggsci> send extract <进程名> , showtrans [thread n] [count n]
```

其中，<进程名>为所要查看的进程名，如 extsz/extxm/extjx 等；

Thread n 是可选的，表示只查看其中一个节点上的未提交交易；

Count n 也是可选的，表示只显示 n 条记录。

例如，查看 extsa 进程中节点 1 上最长的 10 个交易，可以通过下列命令：

示例 8-3：

```
Ggsci> send extract extsa, showtrans thread 1 count 10
```

输出结果是以时间降序排列的所有未提交交易列表，通过 xid 可以查找到对应的事务，请应用开发商和 DBA 帮助可以查找出未提交原因，通过数据库予以提交或者回滚后 GoldenGate 的 checkpoint 会自动向前滚动。

（2）使用 GoldenGate 命令跳过或接受长交易的方法。

在 GoldenGate 中强制提交或者回滚指定事务，可以通过以下命令（<>中的为参数）：

示例 8-4：

```
Ggsci> SEND EXTRACT <进程名>, SKIPTRANS <5.17.27634> THREAD <2>
```

该命令跳过交易。

示例 8-5：

```
Ggsci> SEND EXTRACT <进程名>, FORCETRANS <5.17.27634> THREAD <1>
```

该命令强制认为该交易已经提交。

说明：使用这些命令只会让 GoldenGate 进程跳过或者认为该交易已经提交，但并不改变数据库中的交易，它们依旧存在于数据库中。因此，强烈建议使用数据库中提交或者回滚交易而不是使用 GoldenGate 处理。

（3）配置长交易告警。

可以在 Extract 进程中配置长交易告警，参数如下所示：

示例 8-6：

```
extract extsz
……
warnlongtrans 12h, checkintervals 10m
```

以上表示 GoldenGate 会每隔 10 分钟检查一下长交易，如果有超过 12 个小时的长交易，GoldenGate 会在根目录下的 ggserr.log 里面加入一条告警信息。可以通过查看 ggserr.log 或者在 GGSCI 中执行 view ggsevt 命令查看这些告警信息。

以上配置可以有助于及时发现长交易并予以处理。

说 明

在 OGG 11g 中，Extract 提供了 BR 参数可以设置每隔一段时间（默认 4 小时）将长交易缓存到本地硬盘（默认 dirtmp 目录下），因此 Extract 只要不停止一般需要的归档日志不超过 8 个小时（极限情况）。但是如果 Extract 停掉后，便无法再自动缓存长交易，需要的归档日志就会依赖于停机时间。

8.2　源端和目标端增减复制表

8.2.1　增加复制表

当 GoldenGate 仅打开了 DML 复制时，源端增加复制表的操作步骤为：在 GoldenGate 的进程参数中，如果通过*来匹配所有表，因此只要符合*所匹配的条件，那么只要在源端建立了表之后 GoldenGate 就能自动复制，无需修改配置文件，但是需要为新增的表添加附加日志。

步骤如下：

示例 8-7：

```
GGSCI 〉 dblogin userid GoldenGate, password xxx
GGSCI > info trandata <schema>.<table name>
```

如果不是 enable 则需要手动加入：

示例 8-8：

```
GGSCI > add trandata <schema>.<table name>
```

注：（仅对 Oracle 9i）如果该表有主键或者该表不超过 32 列，则显示 enabled 表示添加成功；如果无主键并且列超过 32 列，则可能出现错误显示无法添加则需要手工处理，此时请根据此方法来处理：

示例 8-9：

```
Alter table <table_name> add supplemental log group <group> (column…)
always;
```

如果没有使用通配符，则需要在主 Extract、Data Pump 里面最后的 table 列表里加入新的复制表；在目标端 Replicat 的 MAP 列表同样也加入该表的映射。

然后，新增表请首先在目标端建立表结构。

如果有外键和 trigger，需要在目标表临时禁止该外键和 trigger，并维护在 dirsql 下的禁止和启用这些对象的对应脚本文件。

对于修改了文件的所有源和目标进程，均需重启进程使新的参数生效。

8.2.2 修改数据表的结构

当数据库需要复制的表结构有所改变，如增加列，改变某些列的属性如长度等可以按照下列步骤执行：

（1）按照本文前面所述操作顺序停止源和目标端各抽取及投递进程（注意停源端抽取要验证一下归档日志是否存在防止无法重起），无需停止 manager 进程。

（2）修改目标表结构。

（3）修改源表结构。

（4）如果表有主键，并且本次修改未修改主键，则可以直接启动源和目标所有进程继续复制，完成本次修改；否则，如果表无主键或者本次修改了主键则需继续执行下列步骤：

示例 8-10：

```
GGSCI> dblogin userid GoldenGate, password xxxx
GGSCI> delete trandata schema.mytable
GGSCI> add  trandata schema.mytable
```

（仅对 Oracle 9i）如果表超过了 32 列则上述操作可能会报错，此时需要手工进行处理，请参考上一节内容。

（5）重新启动源端和目标端的抓取和复制进程。

8.2.3 （仅复制 DML 时）客户应用的升级

如果是客户的应用进行了升级，导致了源系统表的变化，在不配置 DDL 复制的情况下，需要对 GoldenGate 同步进程进行修改，可以参照以下步骤。

（1）停止源和目标端各抽取及投递进程（注意停止源端抽取要验证一下归档日志是否

存在，防止无法重起），无需停止 manager 进程。

（2）对源系统进行升级。

（3）在目标端将客户升级应用所创立的存储过程、表、function 等操作再重新构建一遍。

（4）在目标端手工禁止建立的 trigger 和外键，并将这些 SQL 以及反向维护的（即重新启用 trigger 和外键）SQL 添加到目标端 OGG dirsql 目录下对应的脚本文件里。

> **注意**
>
> 在安装实施时，应当将执行的禁止 trigger 和外键的表放到目标 dirsql 下，文件名建议为 disableTrigger.sql 和 disableFK.sql。同时，需要准备一个反向维护（即重新启用 trigger 和外键，建议为 enableTrigger.sql 和 enableFK.sql）SQL，同样放置到目标端 OGG 的 dirsql 目录下，以备将来接管应用时重新启用。

对于升级过程中在源端增加的表，需要为新增的表添加附加日志。

步骤如下：

示例 8-11：

```
GGSCI 〉 dblogin userid GoldenGate, password ********
GGSCI > info trandata <schema>.<table name>
```

如果不是 enable 则需要手动加入：

示例 8-12：

```
GGSCI > add trandata <schema>.<table name>
```

> **注意**
>
> （仅对 Oracle 9i）如果该表有主键或者该表不超过 32 列，则显示 enabled 表示添加成功；如果无主键并且列超过 32 列，则可能出现错误显示无法添加则需要手工处理，此时请参照上一小节内容处理。

（5）对于升级过程中可以在源端 drop 掉的表，GoldenGate 默认复制所有符合通配符条件的表，可以直接在目标端 drop 掉，无需对复制做任何修改。

（6）如果升级过程中修改了主键的表则需继续执行下列步骤。

示例 8-13：

```
GGSCI> dblogin userid GoldenGate, password xxxx
GGSCI> delete trandata schema.mytable
GGSCI> add  trandata schema.mytable
```

（仅对 Oracle 9i）如果表超过了 32 列则上述操作可能会报错，此时需要手工进行处理，请参考上一小节内容处理。

（7）重新启动源端和目标端的抓取和复制进程。

8.2.4　减少复制表

GoldenGate 默认复制所有符合通配符条件的表，如果有的表不再需要，可以在源端 drop

掉，然后到目标 drop 掉，无需对复制做任何修改。

如果其中几个表依然存在，只是无需 GoldenGate 复制，则可以通过以下步骤排除。

（1）在源端系统上首先验证所需归档日志存在后通过 stop extXX 停止对应的 extXX 进程。

（2）在目标端系统上 GGSCI 中执行 stop repXX 停止目标端的复制进程。

（3）在源端修改 ext 进程的参数文件排除所不复制的表。

在文件定义 table 的行前面加入一行：

示例 8-14：

```
tableexclude <schema>.<tablename>;
```

注意写全 schema 和表的名称。

注：如果是没有使用通配符，则直接注释掉该表所在的 table 行即可。

例如：

示例 8-15：

```
Ggsci> edit param extXX
……
tableexclude ctais2.TMP_*;
tableexclude ctais2.BAK_*;
tableexclude ctais2.MLOG$_*;
tableexclude ctais2.RUPD$_*;
tableexclude ctais2.KJ_*;

tableexclude myschema.mytable;

table ctais2.*;
……
```

（4）在目标端修改 rep 进程参数，同样排除该表：

示例 8-16：

```
GGSCI>edit param repXX
```

在 MAP 前面加入一行：

示例 8-17：

```
--mapexclude CTAIS2.SHOULIXINXI
mapexclude myschema.mytable
MAP ctais2.* ,TARGET ctais2.*;
```

注：如果是没有使用通配符，则直接注释掉该表所在的 MAP 行即可。

（5）在目标端系统上启动复制进程 repXX：

示例 8-18：

```
GGSCI > start  repXX
```

（6）在源端系统上启动源端的抓取进程 extXX：

示例 8-19：

```
GGSCI > start  extXX
```

即可进入正常复制状态。

8.3　数据表重新同步

如果是某些表由于各种原因造成两边数据不一致，需要重新进行同步，可以参照以下步骤。

（1）确认需要重新同步的表。

（2）验证源端各个抽取进程，重起所需归档日志存在，然后停止抽取进程 ext*以及 dpe*相关进程。

（3）停止目标端的 rep 进程。

> **注 意**
>
> 步骤（4）～（6）为将源端数据通过 exp/imp 导入到目标端，客户也可以选择其他初始化方式，比如在目标端为源端表建立 dblink，然后通过“create table as select from”的方式初始化目标端表。

（4）从生产端数据库中获得当前的 SCN：

示例 8-20：

```
col current scn for 9999999999999999
select dbms flashback.get system change number current scn from dual;
     CURRENT_SCN
-----------------
      1647598704
```

（5）在源端使用 exp 导出该表或者几张表数据。例如：

示例 8-21：

```
exp ggs/sprite  DIRECT=y buffer=64000000 FLASHBACK_SCN=1647598704 \
CONSTRAINTS=N GRANTS=N TRIGGERS=N \
file=/GoldenGate/backup/new_tab.dmp log=/GoldenGate/backup/new_tab.log \
tables=MW_APP.MWT_UD_KW_SJBGJL_JK_20101223
```

（6）通过 ftp 传输到目标端。

（7）在目标端，使用 imp 导入数据：

示例 8-22：

```
nohup imp ggs/sprite file=/GoldenGate/backup/new_tab.dmp log=/GoldenGate/
backup/new tab.log fromuser=MW APP touser=MW APP commit=y ignore=y buffer=
52428800 &
```

（8）如果这些表有外键，在目标端检查这些外键并禁止它们（记得维护 dirsql 下的禁止和启用外键的脚本 SQL）。

（9）启动源端和目标端的所有 GoldenGate 进程。

（10）使用 Info all 在 GGSCI 中查看各个进程状况。

8.4 给数据库打补丁

这里说的数据库补丁指的是小补丁，而非数据库大版本的升级。厂商提供给用户的软件补丁的形式多为编译后的库函数。

以 Oracle 为例这里说的补丁主要是指 oneoff patch、PSU 以及 PSU。这些补丁提供的关键的库函数不会变化，所以在 GoldenGate 这个层面无需进行变更设置。比如数据库的版本的 Oracle 10.2.0.4，如果需要给数据库应用最新的 PSU，则 GoldenGate 本身无需变更。但是如果是从 9i 升级到 10g，则对应的 GoldenGate 也需要进行变更。

8.5 给 GoldenGate 程序打补丁

在 GoldenGate 概念里，打补丁等同于升级。升级的一般步骤如下。

（1）待源端与目标端追平且没有延时时，先停止抽取进程，接着停止投递，复制进程。

（2）将新的补丁包程序解压到一个新的目录，然后覆盖掉原来的 GoldenGate 所在目录的文件。

（3）重新启动 GoldenGate 各进程：start replicat **;　start dpe***; start extract ****。

第 3 篇

提　高　篇

第 9 章　GoldenGate 错误分析与处理

在维护 GoldenGate 过程中，由于各种意外情况，难免还是会遇到各种各样的问题。掌握一些常见的 GoldenGate 故障诊断和错误分析的方法是非常有必要的，而且掌握这些错误分析工具也进一步加深对 GoldenGate 产品的认识与对 GoldenGate 原理的理解。

9.1　GoldenGate 常见异常处理

GoldenGate 运行起来后，随着时间的推移可能会碰到各种各样的问题，下面就来介绍常见的异常现象以及常见的异常处理方法。

9.1.1　异常处理的一般步骤

首先确定是 GoldenGate 的哪类进程有故障（是抽取，投递还是复制进程有问题），解决故障的一般思路如下。

（1）通过 GGSCI>view report 命令查找 ERROR 字样，确定错误原因并根据其信息进行排除。

（2）通过 GGSCI>view ggsevt 查看告警日志信息。

（3）检查两端数据库是否正常运行，网络是否连通。

（4）通过 logdump 工具对队列文件进行分析。

9.1.2　RAC 单节点失败

在 RAC 环境下，GoldenGate 软件安装在共享目录下，可以通过任一个节点连接到共享目录，启动 GoldenGate 运行界面。如果其中一个节点失败，导致 GoldenGate 进程中止，可直接切换到另外一个节点继续运行。

操作步骤如下。

（1）以 Oracle 用户登录源系统（使用另外一个正常的节点）。

（2）确认将 GoldenGate 安装的所在文件系统装载到另一节点相同目录。

（3）确认 GoldenGate 安装目录属于 Oracle 用户及其所在组。

（4）确认 Oracle 用户及其所在组对 GoldenGate 安装目录拥有读写权限。

（5）进入 GoldenGate 安装目录。

（6）执行./ggsci 进入命令行界面。

（7）执行 start mgr 启动 MGR。

（8）执行 start er *启动所有进程。

检查各进程是否正常启动，即可进入正常复制。

9.1.3　Extract 常见异常

以下为列举的一些常见错误信息作参考用。

Extract 进程包括抽取与投递进程，投递进程报错大部分原因是由于网络故障。对于源数据库，抽取进程 ext**如果变为 abended，则可以通过在 GGSCI 中使用 view report 命令查看报告，可以通过搜索 ERROR 快速定位错误。

一般情况下，抽取异常的原因是因为其无法找到对应的归档日志，可以通过到归档日志目录命令行下执行

示例 9-1：

```
ls -lt arch_x_xxxx.arc
```

查看该日志是否存在，如不存在则可能的原因如下。

- 日志已经被压缩。
- GoldenGate 无法自动解压缩，需要人工解压缩后才能读取。
- 日志已经被删除。

如果日志已经被删除，需要进行恢复才能继续复制。

一般需要定期备份归档日志，并清除旧的归档日志。需要保证归档日志在归档目录中保留足够长时间之后，才能被备份和清除。即定期备份清除若干小时之前的归档，而不是全部归档。保留时间计算如下。

某归档文件保留时间≥抽取进程处理完该文件中所有日志所需的时间。

可以通过命令行或者 GoldenGate Director Web 界面，运行 info extxx showch 命令查看抓取进程 ext 处理到哪条日志序列号。在此序列号之前的归档，都可以被安全的清除。

抽取进程在抽取不支持的数据对象时也会 abend，report 文件会有详细的报错信息，根据 report 文件来定位错误信息然后再排错即可。

下面再单独列出更多的几个故障。

（1）Extract: Application failded to initialize（Win）。

错误信息：run GGSCI command but the Alert window report "Application failded to initialize(0xc000026e)"。

GoldenGate 在 Windows 平台上需要安装 Microsoft Visual C ++ 2005 SP1 Redistributable Package。如果是 Microsoft Itanium 平台，需要安装 vcredist_IA64.exe。

Windows 2008 需以下额外操作：右击‘cmd’ (DOS)，选择‘run as administrator’，然后在该命令行窗口中启动 MGR 和 Extract 才能够读取数据库日志。

将 OGG 安装为服务时（即运行“install ADDSERVICE”），需要使用管理员权限，这样启动服务后即能访问日志。

通过以下方法为运行 MGR 和 Extract 的用户添加读取日志文件的权限，右键单击文件->property->security->edit->add。

（2）Extract: Cannot load program./ggsci…

错误分析：请首先检查该 OGG Build 是否与操作系统和数据库相符；其次如果是 Aix 请检查 xLC 版本是否符合 10.0 以上。

另外，检查环境变量中动态库路径是否包含了数据库动态库目录，例如：

示例 9-2：

```
export LD_LIBRARY_PATH=$ORACLE_HOME/lib
```

不同平台下的环境变量不同。

- **AIX**　LIBPATH。
- **Solaris、Linux 等**　LD_LIBRARY_PATH。
- **HP-Unix**　SHLIB_PATH。

重设环境变量需重启 Mgr 和 Ext/Rep 进程。

（3）Extract: Block size mismatch (8192/512)…

裸设备的偏移量各操作系统默认为 0，但 AIX 默认为 4096。当创建裸设备时使用了-TO 选项时，Oracle 不会跳过 4096 字节而是直接从 0 开始读写。因此在 AIX 下使用裸设备时，出现此错误需要指定 OGG 从偏移量 0 开始读取。

示例 9-3：

```
tranlogoptions rawdeviceoffset 0
```

该参数其在实际环境中使用几率非常高，在以前版本中如果缺少此参数 Extract 立即终止，但新版本 Extract 会持续进行尝试，并不自动终止，需检查报告文件。

（4）Extract: ORA-15000 ASM connection error

该错误为 OCI 错误，表示 Extract 是在连接数据库时出现问题，根据错误信息判断为权限问题。

首先在 Extract 参数中检查 ASM 相关参数 tranlogoptions asmuser sys@+ASM1，asmpassword oracle，再检查 tnsnames.ora 和 listener.ora 验证 ASM 实例配置是否正确，确认 ASM 用户具有 SYSDBA 权限；如果使用 SYS，需要将 ASM 实例的 init.ora 中 REMOTE_LOGIN_PASSWORDFILE 参数设置为 SHARED（多个数据库可以使用一个 password 文件，只有 SYS 用户可以远程登录）。

使用 sqlplus 验证：

示例 9-4：

```
sqlplus sys/oracle@asm1 as sysdba; //可以登录
sqlplus sys/oracle@asm1;           //报告 15000 错误
```

（5）Extract: Encountered SCN That Is Not Greater Than The Highest SCN Already Processed…

原因分析：在 Oracle RAC 环境中，Extract 会启动一个 coordinator 线程对各个节点上的操作进行根据 SCN 进行排序，它在交易提交后会等待 THREADOPTIONS MAXCOMMITPROPAGATIONDELAY 参数所定义时间来确认空闲节点没有交易，然后再收集交易数据；写入该交易后如果空闲节点后来又读到了一个 SCN 号要小的交易，则会报

告该错误。

可能原因：

- 各节点之间没有配置时钟同步。
- 一个节点比另外一个节点慢（IO 问题可能性较大）。

解决办法：

调整 Extract 参数：

示例 9-5：

```
THREADOPTIONS MAXCOMMITPROPAGATIONDELAY <msec> IOLATENCY <msec>
```

MAXCOMMITPROPAGATIONDELAY 有效范围是 0-90000ms，默认为 3s（即 3000ms）。

GGS V9.x 多了一个 IOLATENCY 参数，可以与上面参数一起加大等待时间。IOLATENCY 默认为 1.5s，最大值为 180000。

建议出现该错误后可以将此二参数设置为较大值，然后逐步降低获取最佳设置。

需要补充说明的是，出现此错误后，因后面的交易可能已被写入日志，重启 Extract 可成功启动，但是可能出现如下问题：Extract 会重写当前队列覆盖前面的交易数据，后面的 Data Pump 进程可能会出现“abend with incompatible record errors”错误终止（旧版本可能出现）。

此问题的恢复步骤如下。

① 停止所有 Data Pump 和 Replicat，针对所有的 Extract 记录其 Write Checkpoint 的队列 Seqno。

② 对于每个 Extract 向下滚动一个队列：

示例 9-6：

```
ALTER EXTRACT [name], ETROLLOVER
```

启动 Extract 查看是否滚动到了下一个队列，记录其新队列 seqno，应当是旧队列号+1。

③ 修改 Data Pump 从新的队列开始传输：

示例 9-7：

```
ALTER EXTRACT [pump_name], EXTSEQNO ##### EXTRBA 0
```

重启 Data Pump 查看是否能够重启成功并从新的队列传输。

④ 修改 Replicat 参数文件，加入或者打开 HANDLECOLLISIONS，如果有 GROUPTRANSOPS 和 MAXTRANSOPS 请注释掉，启动 Replicat，观察其是否能够读取新传输过来的队列如 Replicat 无法自动滚动到下一个队列，需要通过如下命令手工滚动：

示例 9-8：

```
alter replicat [replicat_name], EXTSEQNO ##### EXTRBA 0
```

等待 Replicat 处理到结尾没有延迟时，可以关闭 HANDLECOLLISIONS 和恢复原来的 GROUPTRANSOPS 和 MAXTRANSOPS 参数。

⑤ 重新启动 Replicat 即可恢复正常复制。

9.1.4 网络故障

如果 MGR 进程参数文件里面设置了 autorestart 参数，GoldenGate 可以自动重启，无需人工干预。

当网络不稳定或者发生中断时，GoldenGate 负责产生远地队列的 Pump 进程会自动停止。 此时，MGR 进程会定期根据 mgr.prm 里面 autorestart 设置自动启动 Pump 进程以试探网络是否恢复。在网络恢复后，负责产生远程队列的 Pump 进程会被重新启动，GoldenGate 的检查点机制可以保证进程继续从上次中止复制的日志位置继续复制。

需要注意的是，因为源端的抽取进程（Capture）仍然在不断地抓取日志并写入本地队列文件，但是 Pump 进程不能及时把本地队列搬动到远地，所以本地队列文件无法被自动清除而堆积下来，需要保证足够容量的存储空间来存储堆积的队列文件。计算公式如下。

存储容量≥单位时间产生的队列大小×网络故障恢复时间

MGR 定期启动抓取和复制进程参数配置参考：

示例 9-9：

```
GGSCI > edit param mgr
port 7809
autorestart er *,waitminutes 3,retries 5,RESETMINUTES 60
```

每 3 分钟重试一次，5 次重试失败以后等待 60 分钟，然后重新试三次。

9.1.5 Replicat 进程常见异常

对于目标数据库，投递进程 repXX 如果变为 abended，则可以通过在 GGSCI 中使用 view report 命令查看报告，可以通过搜索 ERROR 快速定位错误。

复制进程的错误通常为目标数据库错误，比如：

- 数据库临时停机。
- 目标表空间存储空间不够。
- 目标表出现不一致。

可以根据报告查看错误原因，排除后重新启动 rep 进程即可。

需要注意一点：往往容易忽略 UNDO 表空间。如果 DML 语句中包含了大量的 UPDATE 和 DELETE 操作，则目标端 UNDO 的生成速度会很快，有可能填满 UNDO 表空间。

典型错误（数据复制典型错误）如下：

示例 9-10：

```
- SQL error 1403 mapping  2010-02-25 13:20:08  GGS WARNING   218  Oracle
GoldenGate Delivery for Oracle, rep stnd.prm:  SQL error 1403 mapping
HR.MY_EMPLOYEE to HR.MY_EMPLOYEE.
```

可能原因包括以下几个方面。

- 两端结构不一致（异构环境，列和主键不同）。
- 两端有不一致记录。
- 附加日志不全。

可以到 discard 文件中查看具体错误信息，如果为 UPDATE 或者 DELETE 找不到对应记录，并且某几个字段为空，则可认定为缺少了附加日志。

9.2　使用 reperror 进行错误处理

对于 Replicat 进程处理 DML 操作过程中报错时，GoldenGate 提供了一个参数用来控制如何处理 Replicat 进程的报错。这就是本节内容要介绍的 reperror 参数。这个参数能控制大部分的 GoldenGate 错误处理的手段。

如某案例的 Replicat 进程参数如图 9-1 所示。

```
REPLICAT repepma
USERID goldengate, PASSWORD  hnggs
SETENV (NLS_LANG = "AMERICAN_AMERICA.ZHS16GBK")
REPORT AT 00:59
REPORTCOUNT EVERY 30 MINUTES, RATE
REPORTROLLOVER AT 02:00
REPERROR DEFAULT, ABEND
REPERROR (1, ignore)
NUMFILES 5000
GROUPTRANSOPS 10000
ASSUMETARGETDEFS
DISCARDFILE ./dirrpt/repepma.dsc, APPEND, MEGABYTES 5200
DISCARDROLLOVER AT 02:00
GETTRUNCATES
ALLOWNOOPUPDATES
CHECKSEQUENCEVALUE
MAP EPM_HA.*, TARGET  EPM_HA.*;
```

图 9-1

9.2.1　reperror 处理类型与含义

Reperror 在 GoldenGate11 版本中共提供了 7 类处理错误方式，分别如下。

（1）abend：Replicat 遇到不能处理的记录时，回滚事务，然后停止处理，Replicat 进程状态转为 abend。

（2）discard：将不能处理记录的错误信息放到 discard 文件而 Replicat 进程继续处理下面的记录。

（3）exception：将错误按照预先定义好的方式处理。

（4）ignore：将不能处理的记录忽略掉，然后继续处理下面的记录。

（5）retryop [maxretries <n>]：遇到不能处理的记录时，重试 n 次。

（6）transabort [,maxretries <n>][, delay[c]sesc<n>];终止事务处理，将 rba 号指到该事务的开头，也可以指定重试几次。

（7）reset：清除掉所有的 reperror 规则，然后将 reperror 的规则默认为 abend。

在 Replicat 进程的参数中，可以将任意一个处理类型设置为默认，如 reperror、default、abend。

通常，为了保证数据的一致性，都将 reperror 的默认规则设置为 abend。

9.2.2 复制进程常见数据库错误类型与处理方法

在实际的 GoldenGate 系统中，很大一部分 Replicat 错误信息都类似于 ORA 开头的数据库错误（这里以 Oracle 数据库为例）。虽然，通常对于 ORA 错误，需要手动查找数据库的原因，但可以用 reperror 处理一些预知的错误类型，然后再在数据库层面找到错误的原因，手动排除，而不至于导致该进程处理其他正常的表而 abend 掉。

例如：可以忽略掉重复数据的插入而其他类型的报错则 abend。

示例 9-11：

```
Reperror (default, abend)
Reperror (-1, ignore)
```

当然，也可以只针对某张表的忽略掉重复数据的插入而 abend 掉其他类型的报错。

示例 9-12：

```
REPERROR (-1, IGNORE)
MAP sales.product, TARGET sales.product;
REPERROR RESET
MAP sales.account, TARGET sales.account;
```

最常见的错误为 ORA-1403。

1403 错误是指记录无法投递到目标库，纯属数据错误，要通过查看错误信息和 discard 文件，到两端库寻找相应记录，结合 logdump 分析队列中的实际数据，再分析出问题的原因。可能存在的原因有：两端表结构不一致；附加日志错误；初始化方法错误导致不一致；目标端级联删除、trigger 没有被禁止；目标端存在 Oracle 的 job 或者操作系统任务修改数据。

处理方法：

- 重新初始化该表。
- 手工修复该条数据。
- 修改 reperror 参数为 discard 或 ignore 模式，忽略掉错误（在使用这个参数之前用户应该非常清楚自己在做什么，因为它会导致两端数据不一致）。

9.3 Ddlerror 处理 DDL 复制错误

当 GoldenGate 打开了 DDL 复制时，当 DDL 复制报错时，则需要用到此处的 ddlerror

参数预处理一些常见的报错信息。Ddlerror 对于抽取、复制进程均有效，默认为 abend。

Ddlerror 参数的语法为：

示例 9-13：

```
DDLERROR
{<error> | DEFAULT} {<response>}
[RETRYOP MAXRETRIES <n> [RETRYDELAY <delay>]]
{INCLUDE <inclusion clause> | EXCLUDE <exclusion clause>}
[IGNOREMISSINGTABLES | ABENDONMISSINGTABLES]
```

如当 DDL 复制报 ORA-1430 错误，传递了重复的 alter 语句导致，则可以用 ddlerror (1430, discard)将错误信息扔到 discard 文件里。

其他的错误处理与 reperror 类似。

9.4　Discardfile 记录进程错误信息

用 discardfile 这个参数来生成一个 discard 文件，将 GoldenGate 不能处理的信息记录到这个文件。这样对 GoldenGate 的 troubleshooting 非常的有帮助。

如源端表结构有变化，默认传递过来的数据应用时 Replicat 进程则报错，此时则可以通过 discard 文件看到报错信息位哪个表做了怎样的 alter 操作，再在目标端也将表结构改变一些，错误即可排除。

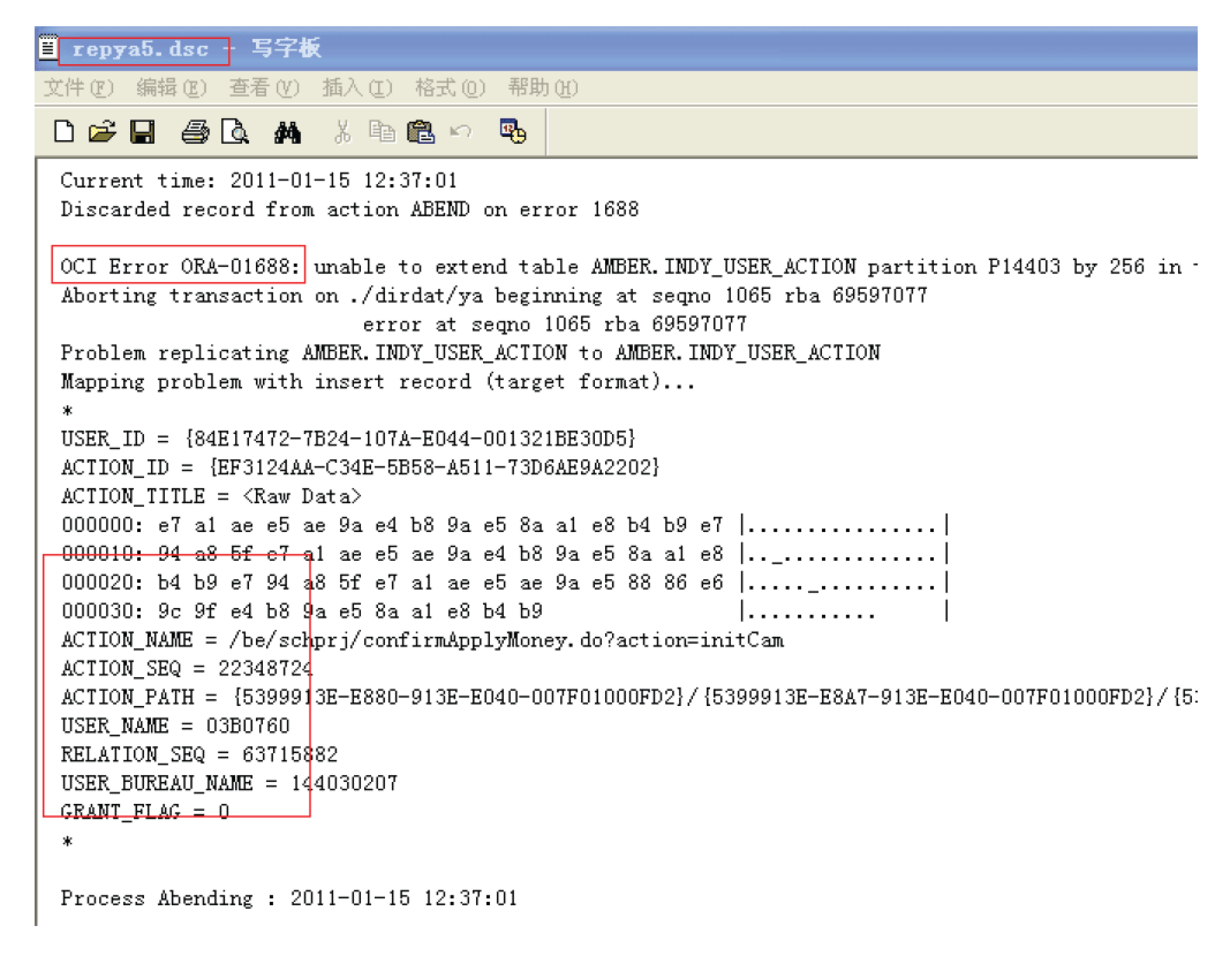
Discard 文件默认在 GoldenGate 安装目录的 dirrpt 子文件夹，如图 9-2 所示。

```
C:\gg\dirrpt 的目录

2011-03-22  13:27    <DIR>          .
2011-03-22  13:27    <DIR>          ..
2011-03-22  13:32             2,552 DPEKFA.rpt
2011-03-22  13:26             2,662 DPEKFA0.rpt
2011-03-22  09:35             3,184 DPEKFA1.rpt
2011-03-22  09:21             3,185 DPEKFA2.rpt
2011-03-22  09:08             3,185 DPEKFA3.rpt
2011-03-22  08:56             3,185 DPEKFA4.rpt
2011-03-22  08:43             3,185 DPEKFA5.rpt
2011-03-21  16:33             3,185 DPEKFA6.rpt
2011-03-21  16:20             3,185 DPEKFA7.rpt
2011-03-21  16:07             3,184 DPEKFA8.rpt
2011-03-21  15:54             3,185 DPEKFA9.rpt
2011-02-25  11:52               503 ext_kfa.dsc
2011-03-22  13:28             1,598 EXT_KFA.rpt
```

图 9-2

Discard 文件记录的报错信息如图 9-3 所示。




```
Current time: 2011-01-15 12:37:01
Discarded record from action ABEND on error 1688

OCI Error ORA-01688: unable to extend table AMBER.INDY_USER_ACTION partition P14403 by 256 in -
Aborting transaction on ./dirdat/ya beginning at seqno 1065 rba 69597077
                         error at seqno 1065 rba 69597077
Problem replicating AMBER.INDY_USER_ACTION to AMBER.INDY_USER_ACTION
Mapping problem with insert record (target format)...
*
USER_ID = {84E17472-7B24-107A-E044-001321BE30D5}
ACTION_ID = {EF3124AA-C34E-5B58-A511-73D6AE9A2202}
ACTION_TITLE = <Raw Data>
000000: e7 a1 ae e5 ae 9a e4 b8 9a e5 8a a1 e8 b4 b9 e7 |................|
000010: 94 a8 5f e7 a1 ae e5 ae 9a e4 b8 9a e5 8a a1 e8 |.._.............|
000020: b4 b9 e7 94 a8 5f e7 a1 ae e5 ae 9a e5 88 86 e6 |....._..........|
000030: 9c 9f e4 b8 9a e5 8a a1 e8 b4 b9                |...........     |
ACTION_NAME = /be/schprj/confirmApplyMoney.do?action=initCam
ACTION_SEQ = 22348724
ACTION_PATH = {5399913E-E880-913E-E040-007F01000FD2}/{5399913E-E8A7-913E-E040-007F01000FD2}/{5:
USER_NAME = 03B0760
RELATION_SEQ = 63715882
USER_BUREAU_NAME = 144030207
GRANT_FLAG = 0
*

Process Abending : 2011-01-15 12:37:01
```

图 9-3

9.5 GoldenGate 常见错误分析

（1）解决 GoldenGate 错误的一个关键点就是通过错误分析工具（包括 report 文件，ggserr.log discard 文件 logdump 工具，GGSCI 命令行）确定错误的根源是哪个组件引起的。

- 系统或者网络？
- 数据库报错或者应用报错？
- GoldenGate 安装报错？
- GoldenGate 的某个进程报错？
- GoldenGate 的参数配置文件报错？
- SQL 语句或者存储过程报错？

然后再确定错误的原因，逐个排查。

（2）当 GoldenGate 遇到错误时，则可以借助日志、report 文件找到错误原因，一步一步来排查。一般的错误信息 GoldenGate 都会提示有相应的解决办法。

如下介绍一个错误案例：

通过命令：

示例 9-14：

```
GGSCI>view ggsevt
```

看到的报错信息如图 9-4 所示。

```
2011-03-01 14:58:56  ERROR   OGG-01032  Oracle GoldenGate Capture for Oracle,
 dpeyb.prm:  There is a problem in network communication, a remote file problem,
 encryption keys for target and source do not match (if using ENCRYPT) or an unknown error.
 Length is 3 - 000000: 46 45 00                                      |FE.             |.
```

图 9-4

通过 view report dpeyb 看到的也是类似的信息。

再来观察容灾端复制进程的报错信息为：

示例 9-15：

```
2011-03-02 12:03:37  ERROR   OGG-01028  Incompatible record in ./dirdat/
yb018262, rba 72955479 (getting header).
```

通过 logdump 进入到该 trail 文件查看，如图 9-5 所示。

```
Logdump 577 >ghdr on
Logdump 578 >detail data
Logdump 579 >detail on
Logdump 580 >open ./dirdat/yb018262
Current LogTrail is /goldengate/dirdat/yb018262
Logdump 581 >pos 72955479
Reading forward from RBA 72955479
Logdump 582 >next
Bad record found at (RBA 72955479, format 5.50 Unknown TokenID 00)
 4701 06b6 4800 0038 4504 0041 0652 73ff 02f1 c397  | G...H..8E..A.Rs....
 7916 e8bd 0000 0000 19d0 e630 0000 77d9 0152 0000  | y.........0..w..R..
 0001 5347 504d 5f4f 5554 2e44 4648 4a50 5f4d 4f4e  | ..SGPM_OUT.DFHJP_MON
 5f53 5800 4400 0652 0327 0000 000a 0000 0000 0000  | _SX.D..R.'.........
 021d fae3 0001 0006 0000 0002 3936 0002 0018 0000  | ............96......
 0014 2020 2020 e586 9ce6 9d91 e5b1 85e6 b091 3c31  | ..    ............<1
 6b56 0003 0005 0000 0001 3400 0400 0600 0000 0253  | kV........4........S
 5300 0500 0700 0000 0353 5353 0006 000a 0000 0006  | S........SSS........
 3230 3131 3032 0007 0004 ffff 0000 0008 0019 0000  | 201102..............
 0015 e5ba 94e6 94b6 e794 b5e8 b4b9 e501 5347 504d  | ................SGPM
 5f4f 5554 2e44 4648 4a50 5f4d 4f4e 5f53 5800 4400  | _OUT.DFHJP_MON_SX.D.
 0658 0329 0000 000a 0000 0000 0000 021d fae3 0001  | .X.)................
 0006 0000 0002 3936 0002 0018 0000 0014 2020 2020  | ......96........
```

图 9-5

通过分析推敲等，确认是因为 trail 文件有一条记录已损坏，导致投递进程不识别，不能自动翻滚到下一个 trail 文件，而复制进程也不能自动应用到下一个 trail 文件，Pump 进程通过手动 etrollover，复制进程通过 alter 手动指定到下一个 trail 文件应用，故障即可排除。

9.5.1　AIX GGSCI 无法运行

错误信息：

示例 9-16：

```
Cannot load ICU resource bundle 'ggMessage', error code 2 - No such file
or directory
Cannot load ICU resource bundle 'ggMessage', error code 2 - No such file
or directory
IOT/Abort trap (core dumped)
```

或者 GGSCI 可以启动，但是运行任何命令都报上面的错误。

处理方法：通常使用已有的 mount 点安装 GoldenGate，在 mount 时使用了并发 CIO 参数。新建文件系统，重新 mount，作为 GoldenGate 安装目录。

错误信息：

示例 9-17：

```
$ ./ggsci
exec(): 0509-036 Cannot load program GGSCI because of the following errors:
        0509-130 Symbol resolution failed for GGSCI because:
        0509-136  Symbol  GetCatName  FiPCc (number 158) is not exported
        from dependent module /usr/lib/libC.a[ansi 64.o].
        0509-136  Symbol  Getnumpunct  FPCc (number 162) is not exported
        from dependent module /usr/lib/libC.a[ansi 64.o].
        0509-136  Symbol __ct__Q2_3std8_LocinfoFPCci (number 183) is not
        exported from dependent module /usr/lib/libC.a[ansi_64.o].
        0509-192 Examine .loader section symbols with the 'dump -Tv' command.
```

原因是 XLC 是 6.0 版本，升级 XLC 版本到 10.1 以上，问题即可解决。

9.5.2 HP-UX GGSCI 无法运行

错误信息：core dumped

该问题只在 HP-UX11.31 上发现。

处理方法：环境变量没有设置正确。

9.5.3 OGG-01296

示例 9-18：

```
ERROR   OGG-01296  Oracle GoldenGate Delivery for Oracle, yx rep3.prm:
Error mapping from SGPM.A_PAY_FLOW to SGPM.A_PAY_FLOW.
```

由于源端进行了表结构更改，没有通知目标端，导致此错误。

处理方法：在目标端执行相应的语句，将表结构修改为和源端一致。

9.5.4 OGG-01088

错误信息：

示例 9-19：

```
ERROR   OGG-01088  Oracle GoldenGate Delivery for Oracle, pms rep1.prm:
malloc 2097152 bytes failed.
ERROR   OGG-01668  Oracle GoldenGate Delivery for Oracle, pms_rep1.prm:
PROCESS ABENDING.
```

处理方法：

（1）“ulimit –a”，验证操作系统对用户是否所有资源都是无限制。

（2）将进程进行拆分，拆分为多个进程。

（3）从 support.oracle.com 下载最新的补丁包，升级 GoldenGate。

9.5.5　OGG-01224

示例 9-20：

```
ERROR OGG-01224 Oracle GoldenGate Manager for Oracle, mgr.prm: No buffer
space available
```

处理方法：

修改 mgr.prm，扩大动态端口范围，dynamicportlist 7840-7914。

9.5.6　OGG-01031

示例 9-21：

```
ERROR   OGG-01031  There is a problem in network communication, a remote file
problem, encryption keys for target and source do not match (if using ENCRYPT)
or an unknown error. (Reply received is Expected 4 bytes, but got 0 bytes,
in trail ./dirdat/t1000026, seqno 26, reading record trailer token at RBA
103637218).
2011-01-06 11:04:16  ERROR   OGG-01668  PROCESS ABENDING.
```

处理方法：

可能是目标端的 trail file 出问题了，前滚重新生成一个新的 SEND EXTRACT xxx ROLLOVER，或者“alter extract xxx rollover”。

9.5.7　OGG-01072

示例 9-22：

```
ERROR OGG-01072 LOBROW_get_next_chunk(LOBROW_row_t *, BOOL, BOOL, BOOL,
LOBROW_chunk_header_t *, char *, size_t, BOOL, *) Buffer overflow, needed:
132, alloc 2.
```

处理方法：

（1）如果版本为 11.1.1.0.1 Build 078 版本，升级到最新的补丁包。

（2）使用“ulimit –a”查看资源使用限制，调整资源为 unlimited。

（3）Extract: DBOPTIONS LOBBUFSIZE <bytes>。

（4）replicat: DBOPTIONS LOBWRITESIZE 1MB。

9.5.8 OGG-01476

示例 9-23：

```
ERROR   OGG-01476  The previous run abended due to an out of order transaction. Issue ALTER ETROLLOVER to advance the output trail sequence past the current trail sequence number, then restart. Then, use ALTER EXTSEQNO on the subsequent pump EXTRACT, or REPLICAT, process group to start reading from the new trail file created by ALTER ETROLLOVER; the downstream process will not automatically switch to the new trail file.
```

在初始化的时候，由于容灾端没有准备就绪，生产端来回进行了很多次的操作，导致生产端抽取混乱，此时在进行 RMAN 之前，重新启动抽取，忽略调之前的混乱信息。

处理方法：“alter extract xxx, etrollover”。

9.5.9 OGG-00850

示例 9-24：

```
ERROR   OGG-00850  Oracle GoldenGate Capture for DB2, extxa.prm:  Database instance XP1 has both USEREXIT and LOGRETAIN set to off.
ERROR   OGG-01668  Oracle GoldenGate Capture for DB2, extxa.prm:  PROCESS ABENDING.
```

处理方法：

（1）如果是 DB2 8.1/8.2，必须将 USEREXIT 和 LOGRETAIN 设置为 ON。

（2）如果是 DB2 9.5，已经使用 LOGARCHMETH1 和 LOGARCHMETH2 代替以上两个参数，通常 LOGARCHMETH1 为 DISK，LOGARCHMETH2 为 TSM，采用这两个参数开启归档模式。在 DB2 9.5 中，USEREXIT 可以设置为 OFF，但是 LOGRETAIN 仍需设置为 ON。

9.5.10 OGG-01027（长事务）

示例 9-25：

```
WARNING OGG-01027  Long Running Transaction: XID 82.4.242063, Items 0, Extract YX_EXT1, Redo Thread 1, SCN 2379.2132775890 (10219859973074), Redo Seq #5688, Redo RBA 195997712.
```

可以通过下面的命令寻找更详细的信息：

示例 9-26：

```
GGSCI> send extract xxx, showtrans [thread n] [count n]
```

其中，thread n 是可选的，表示只查看其中一个节点上的未提交交易；count n 也是可

选的，表示只显示 n 条记录。

例如查看 xxx 进程中节点 1 上最长的 10 个交易，可以通过下列命令：

示例 9-27：

```
GGSCI> send extract extsz , showtrans thread 1 count 10
```

记录 XID，通过 DBA 查找具体的长交易执行的内容：

示例 9-28：

```
GGSCI> SEND EXTRACT xxx, SKIPTRANS <82.4.242063> THREAD <2> //跳过交易
GGSCI>SEND EXTRACT xxx, FORCETRANS <82.4.242063> THREAD <1> //强制认为该交
                                                               易已经提交
```

使用这些命令只会让 GoldenGate 进程跳过或者认为该交易已经提交，但并不改变数据库中的交易，它们依旧存在于数据库中。因此，强烈建议使用数据库中提交或者回滚交易而不是使用 GoldenGate 处理。

9.5.11 队列文件保存天数

在 mgr.prm 中，添加：

示例 9-29：

```
PURGEOLDEXTRACTS ./dirdat/*,usecheckpoints, minkeepdays 3
```

修改之后，必须重启 manager 即可看到队列文件占用的空间被按照上面指定的规则释放。

如果存储空间不够，可以将 minkeepdays 修改为 MINKEEPHOURS。

如果源端存储空间不足，最好修改最少保留的时间。

9.5.12 复制进程拆分及指定队列文件及 RBA

拆分前通过 INFO XXX 获取队列文件信息及 RBA 号，返回样例如下：

示例 9-30：

```
GGSCI> INFO REPYXA
REPLICAT   REPYXA    Last Started 2011-01-08 19:48   Status RUNNING
Checkpoint Lag       00:00:00 (updated 00:01:42 ago)
Log Read Checkpoint  File ./dirdat/p1000556 First Record  RBA 59193235
```

在将 Replicat 进程拆分后，指定从拆分前的队列文件及 RBA 号码开始复制：

示例 9-31：

```
ALTER REPLICAT xxx EXTSEQNO nnn, EXTRBA mmm
```

以上面的为例：

示例 9-32：

```
ALTER REPLICAT REPYXA 556, EXTRBA 59193235
```

9.5.13 BOUNDED RECOVERY

错误信息：

示例 9-33：

```
BOUNDED RECOVERY: reset to initial or altered checkpoint.
```

数据库问题，不能读取第 2 个节点的 archivelog 文件。

9.5.14 排除不复制的表

在参数文件中增加：

示例 9-34：

```
TABLEEXCLUDE schema.table_name
```

9.5.15 从指定时间重新抓取

重新抓取数据前提：归档文件没有删除。

示例 9-35：

```
ALTER EXTRACT xxx, TRANLOG, BEGIN 2010-12-31 08:00
```

时间格式：yyyy-mm-dd [hh:mi:[ss[.cccccc]]]

如果是新建：

示例 9-36：

```
ADD EXTRACT xxx, TRANLOG, BEGIN 2010-12-31 08:00
```

9.5.16 进程无法停止

通常情况是在处理大交易，尤其在有超过 2 小时以上的大交易，建议等待进程处理完毕。

处理方法：如果必须停止进程，可以强制杀死进程。

示例 9-37：

```
send xxx forcestop
```

9.5.17 CLOB 处理

如果包含 CLOB 字段，在 Extract 参数文件中必须添加：

示例 9-38：

```
TRANLOGOPTIONS CONVERTUCS2CLOBS
```

9.5.18 DB2 不能使用 checkpoint table

处理方法：在增加 Replicat 进程时使用 nodbcheckpoint 参数。

示例 9-39：

```
add replicat xxx, exttrail /GoldenGate/dirdat/rb, nodbcheckpoint
```

9.6 中文表/中文字段处理

比如有个如下的中文表：

示例 9-40：

```
create table 测试表(
ID NUMBER,
姓名 VARCHAR2(30),
FLAG CHAR(1),
CONSTRAINT PK TESTD PRIMARY KEY (ID) USING INDEX);

--源端创建 MV LOG 和 MV：
drop materialized view log on "测试表";
create materialized view log on "测试表" with primary key;
drop materialized view mv cn table;
create materialized view mv cn table refresh fast on commit as select id,
姓名 as en_name,flag from "测试表";
```

在目标端创建表及 view：

示例 9-41：

```
create or replace view v cn table as select id,姓名 as en name,flag  from 测
试表;
```

这里 NLS_LANG 在 GG 中，抽取和复制必须设置为和目标字符集一致：

示例 9-42：

```
SETENV (NLS_LANG = "AMERICAN_AMERICA.AL32UTF8")
```

Extract 相关：

示例 9-43：

```
extract ODISC
SETENV (NLS_LANG = "AMERICAN_AMERICA.AL32UTF8")
```

```
userid custom src, password custom src
exttrail D:/GoldenGate/dirdat/ODISoc/oc
TABLE CUSTOM_SRC.MV_CN_TABLE;
```

Pump 相关：

示例 9-44：

```
extract ODIT1P
SETENV (NLS LANG = "AMERICAN AMERICA.AL32UTF8")
PASSTHRU
rmthost localhost, mgrport 7909
rmttrail D:/gg stg/dirdat/ODIT1op/op
TABLE CUSTOM_SRC.MV_CN_TABLE;
```

Replicat 相关：

示例 9-45：

```
replicat ODIT1A1
SETENV (NLS LANG = "AMERICAN AMERICA.AL32UTF8")
userid odi_staging,  password odi_staging
discardfile D:/gg_stg/dirrpt/ODIT1.dsc, purge
ASSUMETARGETDEFS
```

这里必须指定 APPLYNOOPUPDATES 参数，否则 UPDATE 有问题，另外，也要指定 KEYCOLS，否则删除和更新有问题：

示例 9-46：

```
map CUSTOM_SRC.MV_CN_TABLE, TARGET ODI_STAGING.V_CN_TABLE, KEYCOLS (ID);
```

9.7 Logdump 分析工具

Logdump 是一个 GoldenGate 自带的 trail 文件分析工具，而且能加深对 GoldenGate 工作原理的理解，非常值得花时间来研究它。

Logdump 组件默认在安装目录，是个可执行文件，如图 9-6 所示。

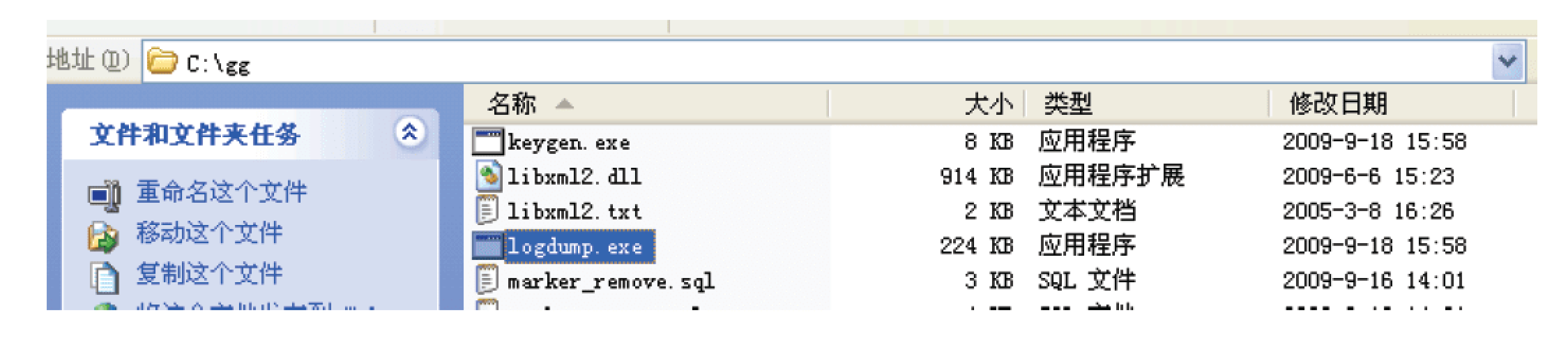


图 9-6

直接输入 logdump 按回车键即可进入 logdump 界面，如图 9-7 所示。

```
C:\gg>logdump

Oracle GoldenGate Log File Dump Utility
Version 10.4.0.19 Build 002

Copyright (C) 1995, 2009, Oracle and/or its affiliates.  All rights reserved.


Logdump 226 >versions
```

图 9-7

9.7.1　认识 logdump 分析工具及常用命令

logdump 的命令不是很多，经常用的就那么几条，进入 logdump 命令行之后，直接可以输入 help 即可看到用的每一条命令，如图 9-8 和图 9-9 所示。

```
Logdump 227 >help

FC [<num> | <string>]      - Edit previous command
HISTORY                    - List previous commands
OPEN | FROM  <filename>    - Open a Log file
RECORD | REC               - Display audit record
NEXT [ <count> ]           - Display next data record
SKIP [ <count> ] [FILTER]  - Skip down <count> records
     FILTER                - Apply filter during skip
COUNT                      - Count the records in the file
      [START[time] <timestr>,]
      [END[time] <timestr>,]
      [INT[erval] <minutes>,]
      [LOG[trail] <wildcard-template>,]
      [FILE <wildcard-template>,]
      [DETAIL ]
       <timestr> format is
         [[yy]yy-mm-dd] [hh[:mm][:ss]]
POSITION [ <rba> | FIRST | LAST | EOF ] - Set position in file
         REVerse | FORward             - Set read direction
RECLEN [ <size> ]  - Sets max output length
EXIT | QUIT        - Exit the program
FILES | FI | DIR   - Display filenames
ENV                - Show current settings
VOLUME | VOL | V   - Change default volume
DEBUG              - Enter the debugger
GHDR  ON | OFF     - Toggle GHDR display
DETAIL ON | OFF | DATA - Toggle detailed data display
RECLEN <nnn>        - Set data display length
SCANFORHEADER (SFH)  [PREV]  - Search for the start of a header
SCANFORTYPE   (SFT) - Find the next record of <TYPE>
      <typename> | <typenumber>
      [,<filename-template>]
SCANFORRBA    (SFR) - Find the next record with <SYSKEY>
      <syskey>              - syskey = -1 scans for next record
      ,<filename-template>
```

图 9-8

```
       GGSTOKEN <tokenname> [<comparison>] [<tokenvalue>]
       USERTOKEN <tokenname> [<comparison>] [<tokenvalue>]
       CSN | LogCSN [<comparison>] [<value>]
   <column range>
       <start column>:<end column>, ie  0:231
   <comparison>
       =, ==, !=, <>, <, >, <=, >=  EQ, GT, LE, GE, LE, NE
{ <program> [string]  - Execute <program>
TRANSHIST nnnn          - Set size of transaction history
TRANSRECLIMIT nnnn      - Set low record count threshold
TRANSBYTELIMIT nnnn     - Set low byte count threshold
LOG <STOP> | { [TO] <filename> } - Write a session log
BEGIN <date-time>        - Set next read position using a timestamp
SAVE <savefilename> [!] <options>  - Write data to a savefile
   <options> are
   nnn RECORDS | nnn BYTES
   [NOCOMMENT]  - Suppress the Comment header/trailer recs, Default
   [COMMENT]    - Insert Comment header/trailer recs
   [OLDFORMAT]  - Force oldformat records
   [NEWFORMAT]  - Force newformat records
   [TRUNCATE ]  - purgedata an existing savefile
   [EXT ( <pri>, <sec> [,<max>])] - Savefile Extent sizes on NSK
   [MEGabytes <nnnn>]             - For extent size calculation
   [TRANSIND <nnn>]               - Set the transind field
   [COMMITTS <nnn>]               - Set the committs field
USERTOKEN      on   | OFF | detail  - Show user token info
HEADERTOKEN    on   | OFF | detail  - Show header token info
GGSTOKEN       on   | OFF | detail  - Show GGS token info
FILEHEADER     on   | OFF | detail  - Display file header contents
ASCIIHEADER    ON   | off           - Toggle header charset
EBCDICHEADER   on   | OFF           - Toggle header charset
ASCIIDATA      ON   | on            - Toggle user data charset
EBCDICDATA     on   | OFF           - Toggle user data charset
ASCIIDUMP      ON   | off           - Toggle charset for hex/ascii display
EBCDICDUMP     on   | OFF           - Toggle charset for hex/ascii display
TRAILFORMAT    old | new            - Force trail type
PRINTMXCOLUMNINFO   on | OFF        - Toggle SQL/MX columninfo display
TMFBEFOREIMAGE      on | OFF        - Toggle display of TMF before images
FLOAT   <value>                     - Interpret a floating point number
       [FORMAT <specifier>]         - sprintf format default %f
```

图 9-9

其中常用的命令如下。

- **Usertoken** 用来显示 trail 文件的一些标记信息。
- **Ggstoken** 显示 GoldenGate 的一些标记信息。
- **Headertoken** 显示头部的标记信息。
- **ghdr on** 用来打开记录的头部信息。
- **pos first|last| for| rev** 其中 pos for|rev 比较重要，用于指定在 trail 文件中读的方向（向上|向下）。
- **count detail** 会显示当前 trail 文件总的记录数量，如图 9-10 所示。

其他一些命令查看文档很容易就能明白。

```
LogTrail C:\gg\yb018262 has 179855 records
Total Data Bytes          54476265
  Avg Bytes/Record             302
Delete                       86874
Insert                       87799
FieldComp                     1899
GGSPKUpdate                   3281
RestartAbend                     1
Others                           1
Before Images                86874
After Images                 92980

Average of 2274 Transactions
    Bytes/Trans .....      27752
    Records/Trans ...         79
    Files/Trans .....          2
```

图 9-10

9.7.2　理解 trail 文件格式与常见分析思路

理解 trail 文件的格式对理解 GoldenGate 的工作原理非常重要，对 troubleshooting 也非常的有帮助，如图 9-11 所示。

```
Logdump 237 >n
TokenID x46 'F' Record Header    Info xff80  Length 1339
TokenID x30 '0' TrailInfo        Info x00  Length  560
TokenID x31 '1' MachineInfo      Info x00  Length   55
TokenID x32 '2' DatabaseInfo     Info x00  Length  287
TokenID x33 '3' ProducerInfo     Info x00  Length   82
TokenID x34 '4' ContinuityInfo   Info x00  Length  327
TokenID x5a 'Z' Record Trailer   Info xff80  Length 1339

2011/03/01 14:53:34.880.830 FileHeader          Len   1331 RBA 0
Name: *FileHeader*
 3000 0230 3000 0008 4747 0d0a 544c 0a0d 3100 0002 | 0..00...GG..TL..1...
 0002 3200 0004 2000 0000 3300 0008 02f1 c399 b60d | ..2... ...3.........
 983e 3400 001a 0018 7572 693a 7378 7978 6462 3031 | .>4.....uri:sxyxdb01
 3a3a 676f 6c64 656e 6761 7465 3500 0096 3500 001a | ::goldengate5...5...
 0018 7572 693a 7378 7978 6462 3031 3a3a 676f 6c64 | ..uri:sxyxdb01::gold
 656e 6761 7465 3500 001a 0018 7572 693a 7378 7978 | engate5.....uri:sxyx
 6462 3031 3a3a 676f 6c64 656e 6761 7465 3500 001a | db01::goldengate5...

Logdump 238 >n
TokenID x47 'G' Record Header    Info x01  Length   60
TokenID x48 'H' GHDR             Info x00  Length   35
TokenID x54 'T' GGS Tokens       Info x00  Length    9
TokenID x5a 'Z' Record Trailer   Info x01  Length   60

2011/03/01 14:51:13.939.603 RestartAbend        Len     0 RBA 1339
Name:
After  Image:                                          Partition 0    G  s

GGS tokens:
 4e00 0005 4450 4559 42                          | N...DPEYB
```


图 9-11

其中，一个 trail 文件最先开始是文件的头部信息，然后是记录的头部信息，接着为记

录的详细信息。GoldenGate 是以事务为单位处理记录的，所以 trail 文件记录的时间均为 commit 的时间，如图 9-12 所示。

```
Logdump 239 >n
TokenID x47 'G' Record Header    Info x01  Length  335
TokenID x48 'H' GHDR             Info x00  Length   60
TokenID x44 'D' Data             Info x00  Length  197
TokenID x54 'T' GGS Tokens       Info x00  Length   58
TokenID x5a 'Z' Record Trailer   Info x01  Length  335

2011/03/01 12:12:40.149.766 Delete                    Len   197 RBA 1399
Name: SGPM_OUT.S_RPT_HANDLE_DAY
Before Image:                                               Partition 4   G  b
 0000 0015 0000 3230 3131 2d30 322d 3235 3a32 303a | ......2011-02-25:20:
 3030 3a30 3000 0100 0b00 0000 0731 3434 3038 3037 | 00:00........1440807
 0002 000e 0000 000a 3030 3033 3032 3539 3330 0003 | ........0003025930..
 0007 0000 0003 3030 3100 0400 1600 0000 12e6 9585 | ......001...........
 e99a 9ce5 a484 e790 86e5 aea1 e6a0 b800 0500 0a00 | ....................
 0000 0000 0000 0000 0200 0600 0a00 0000 0000 0000 | ....................
 0000 0200 0700 0a00 0000 0000 0000 0000 0000 0800 | ....................
Column     0 (x0000), Len    21 (x0015)
 0000 3230 3131 2d30 322d 3235 3a32 303a 3030 3a30 | ..2011-02-25:20:00:0
 30                                                 | 0
Column     1 (x0001), Len    11 (x000b)
 0000 0007 3134 3430 3830 37                        | ....1440807
Column     2 (x0002), Len    14 (x000e)
 0000 000a 3030 3033 3032 3539 3330                 | ....0003025930
Column     3 (x0003), Len     7 (x0007)
 0000 0003 3030 31                                  | ....001
Column     4 (x0004), Len    22 (x0016)
 0000 0012 e695 85e9 9a9c e5a4 84e7 9086 e5ae a1e6 | ....................
 a0b8                                               | ..
Column     5 (x0005), Len    10 (x000a)
 0000 0000 0000 0000 0002                           | ..........
Column     6 (x0006), Len    10 (x000a)
 0000 0000 0000 0000 0002                           | ..........
Column     7 (x0007), Len    10 (x000a)
 0000 0000 0000 0000 0000                           | ..........
Column     8 (x0008), Len    10 (x000a)
 0000 0000 0000 0000 0002                           | ..........
Column     9 (x0009), Len    10 (x000a)
```

操作类型，此处为delete

十六进制信息

列值信息

图 9-12

Trial 文件中的处在同一事物中的 record，由 I/O time 和 AuditRBA 值唯一确定，如图 9-13 所示。

```
Hdr-Ind    :     E  (x45)     Partition  :     .  (x04)
UndoFlag   :     .  (x00)     BeforeAfter:     B  (x42)
RecLength  :   186  (x00ba)   IO Time    : 2011/03/01 12:12:40.147.634
IOType     :     3  (x03)     OrigNode   :   255  (xff)
TransInd   :     .  (x01)     FormatType :     R  (x52)
SyskeyLen  :     0  (x00)     Incomplete :     .  (x00)
AuditRBA   :      30681       AuditPos   : 353693624
Continued  :     N  (x00)     RecCount   :     1  (x01)

2011/03/01 12:12:40.147.634 Delete                  Len   186 RBA 89718
Name: SGPM_OUT.S_RPT_HANDLE_DAY
Before Image:                                             Partition 4   G  m
 0000 0015 0000 3230 3131 2d30 322d 3235 3a32 303a | ......2011-02-25:20:
 3330 3a30 3000 0100 0900 0000 0531 3434 3031 0002 | 30:00........14401..
 000b 0000 0007 3330 3235 3739 3900 0300 0700 0000 | ......3025799.......
```

该条记录为当前事务中间的一条记录

图 9-13

TransInd 记录当前的 record 处于当前事务的哪个部分，如图 9-14 所示。

```
TransInd   : .  (x00) 为该事务的第一条记录
TransInd   : .  (x01) 处在该事务的中间
TransInd   : .  (x02) 处在该事务的结尾
TransInd   : .  (x03) 该事务仅包含一条记录
```

图 9-14

GoldenGate 的工作原理可以理解为源端抽取进程从 redo 日志中按事物抽取相应的 DML 操作信息与数据，然后按时间顺序写到本地的 trail 文件，然后源端的投递进程则与容灾端 MGR 进程、Extract 进程交互，通过读 trail 文件信息，把数据通过 TCP/IP 协议投递到容灾端。接着容灾端的 Replicat 进程再通过读投递过来的 trail 文件，转换为 SQL 语句往数据库里更新数据，这样就完整地实现了数据复制。

了解到这一层后，如果 GoldenGate 链路出现问题，或者 trail 文件有损坏，则可以直接通过 logdump 分析到对应的检查点，重新抽取或者投递或者复制，从而可以修复已经损坏的链路。

9.7.3 Logdump 使用指引

在 GGSCI 中使用如下命令查看当前处理的队列文件和 RBA 号，例如：

示例 9-47：

```
GGSCI (br-jl-accs-db1) 3> info REPYXA
REPLICAT   REPYXA    Last Started 2011-01-08 19:48   Status RUNNING
Checkpoint Lag       00:00:00 (updated 136:41:42 ago)
Log Read Checkpoint  File ./dirdat/p1000556 First Record  RBA 59193235
```

在 GoldenGate 安装目录执行 logdump 命令，打开要查看的队列文件：

示例 9-48：

```
Logdump 1 >open ./dirdat/p1000556
Current LogTrail is ./dirdat/p1000556
Logdump 2 >ghdr on
Logdump 3 >detail on
Logdump 4 >detail data
Logdump 5 >usertoken on
Logdump 6 >pos 59193235 上面 INFO 命令看到的 RBA 号码
Logdump 7 >n
```

继续输入 n 显示当前处理的表及相关操作。

第 10 章　GoldenGate 的安全特性

GoldenGate 软件已经被很多大型企业用于数据容灾。如果用作异地备份容灾，很多是需要通过租用公网的线路进行传输，而这些数据很多都是企业的机密，为了防止机密数据被黑客获取进而损害企业的利益，需要对 GoldenGate 的安全做一些增强。

除了通过制定操作系统和数据库级别安全防范措施以外，还可以在 GoldenGate 层面来制定相应的安全策略。在本地可以通过加密 trail 文件和数据库文件来保护 GoldenGate 抽取到的数据。在网络传输过程中 GoldenGate 也可以加密传输的数据，用户可以自己定义 key 来加密数据，使得黑客就算获取了数据也无法对其解密。

下面来一一介绍着几种保护 GoldenGate 和数据安全的方法。

10.1　加密 trail 文件

加密 extract trail 文件非常的简单，只需要在 Extract 参数文件中加入 ENCRYPTTRAIL 参数。Extract 进程就会对加入参数以后生成的 trail 文件进行加密。如果生产端 trail 文件加密，那么在容灾端参数文件中必须加入对应的 DECRYPTTRAIL 参数解密 trail 文件再入库。

下面用 logdump（查看 GoldenGate trial 文件的工具）对比一下加密之前和加密以后 trail 文件中内容的变化。

没加密之前 Extract 的内容：

示例 10-1：

```
GGSCI (OE5) 55> view params extma

EXTRACT extma
userid GoldenGate@orcl1, password GoldenGate
setenv (NLS LANG="AMERICAN AMERICA.WE8ISO8859P1")
GETTRUNCATES
REPORTCOUNT EVERY 1 MINUTES, RATE
numfiles 50000
DISCARDFILE ./dirrpt/extma.dsc,APPEND,MEGABYTES 50
WARNLONGTRANS 2h,CHECKINTERVAL 3m
EXTTRAIL ./dirdat/ma
DBOPTIONS ALLOWUNUSEDCOLUMN
TRANLOGOPTIONS  CONVERTUCS2CLOBS
DYNAMICRESOLUTION
table scott.* ;
```

没加密之前 Extract trail 文件的内容：

示例 10-2：

```
Logdump 55 >open ./dirdat/ma000001
Current LogTrail is /opt/GoldenGate/orcl1/dirdat/ma000001
Logdump 56 >ghdr on
Logdump 57 >detail data
Logdump 58 >ggstoken detail
Logdump 59 >pos 0
Reading forward from RBA 0
Logdump 60 >n

Logdump 65 >n

Hdr-Ind     :     E  (x45)   Partition   :     .  (x04)
UndoFlag    :     .  (x00)   BeforeAfter :     A  (x41)
RecLength   :    23  (x0017) I/O Time    : 2011/03/22 00:09:39.000.000
IOType      :     5  (x05)   OrigNode    :   255  (xff)
TransInd    :     .  (x00)   FormatType  :     R  (x52)
SyskeyLen   :     0  (x00)   Incomplete  :     .  (x00)
AuditRBA    :          2     AuditPos    : 29881732
Continued   :     N  (x00)   RecCount    :     1  (x01)

2011/03/22 00:09:39.000.000 Insert        Len    23 RBA 1391
Name: SCOTT.TEST
After  Image:                             Partition 4   G  b
 0000 0005 0000 0001 3100 0100 0a00 0000 066f 7261 | ........1........ora
 636c 65                                           | cle
Column     0 (x0000), Len     5 (x0005)
 0000 0001 31                                      | ....1
Column     1 (x0001), Len    10 (x000a)
 0000 0006 6f72 6163 6c65                          | ....oracle  --可以明
 显的看到单词

GGS tokens:
TokenID x52 'R' ORAROWID        Info x00  Length   20
 4141 414d 3058 4141 4541 4141 4147 5741 4141 0001 | AAAM0XAAEAAAAGWAAA..
TokenID x4c 'L' LOGCSN          Info x00  Length    6
 3438 3937 3831                                    | 489781
TokenID x36 '6' TRANID          Info x00  Length    8
 392e 3130 2e32 3939                               | 9.10.299
```

接下来再在参数文件中加入 ENCRYPTTRAIL 参数，使其对 trail 文件加密：

示例 10-3：

```
GGSCI (OE5) 55> view params extma
```

```
EXTRACT extma
userid GoldenGate@orcl1, password GoldenGate
setenv (NLS LANG="AMERICAN AMERICA.WE8ISO8859P1")
GETTRUNCATES
REPORTCOUNT EVERY 1 MINUTES, RATE
numfiles 50000
DISCARDFILE ./dirrpt/extma.dsc,APPEND,MEGABYTES 50
WARNLONGTRANS 2h,CHECKINTERVAL 3m
ENCRYPTTRAIL
EXTTRAIL ./dirdat/ma
DBOPTIONS ALLOWUNUSEDCOLUMN
TRANLOGOPTIONS  CONVERTUCS2CLOBS
DYNAMICRESOLUTION
table scott.* ;
```

再查看加密后生成的 Extract trail 文件内容：

示例 10-4：

```
Logdump 66 >open ./dirdat/ma000002
Current LogTrail is /opt/GoldenGate/orcl1/dirdat/ma000002
Logdump 67 >ghdr on
Logdump 68 >detail data
Logdump 69 >ggstoken detail

Logdump 74 >n
___________________________________________________________________
Hdr-Ind     :     E  (x45)  Partition   :     .  (x04)
UndoFlag    :     .  (x00)  BeforeAfter :     A  (x41)
RecLength   :    24  (x0018) I/O Time    : 2011/03/22 00:35:13.000.000
IOType      :     5  (x05)  OrigNode    :   255  (xff)
TransInd    :     .  (x01)  FormatType  :     R  (x52)
SyskeyLen   :     0  (x00)  Incomplete  :     .  (x00)
AuditRBA    :         2     AuditPos    : 31891236
Continued   :     N  (x00)  RecCount    :     1  (x01)

2011/03/22 00:35:13.000.000 Insert             Len    24 RBA 1212
Name: SCOTT.TEST
After  Image:                                  Partition 4   G  m
 5e50 86ba af70 962b cc52 5bf9 a3f7 9760 7eda abd0 | ^P...p.+.R[....`~...
 -加密后看到的是不可识别的密文
 c092 111e                                         | ....
Bad compressed block, found length of 34490 (x86ba), RBA 1212

GGS tokens:
TokenID x52 'R' ORAROWID         Info x00  Length   20
 4141 414d 3058 4141 4541 4141 4147 5741 4130 0001 | AAAM0XAAEAAAAGWAA0..
```

加密后容灾端进程 abend。

下面是容灾端进程的参数和错误信息：

示例 10-5：

```
GGSCI (OE5) 3> view params repma

REPLICAT repma
USERID GoldenGate@orcl2, PASSWORD GoldenGate
setenv (NLS LANG="AMERICAN AMERICA.WE8ISO8859P1")
--REPORT AT 01:59
REPORTCOUNT EVERY 30 MINUTES, RATE
REPERROR DEFAULT, abend
numfiles 50000
DBOPTIONS ALLOWUNUSEDCOLUMN
MAXTRANSOPS   500000
GROUPTRANSOPS 10000
CHECKPOINTSECS 40
--HANDLECOLLISIONS
assumetargetdefs
DISCARDFILE ./dirrpt/repma.dsc, APPEND, MEGABYTES 50
GETTRUNCATES
ALLOWNOOPUPDATES
map scott.* , target scott.* ;

-----ERROR 信息————————
Source Context :
  SourceModule          : [ggstd.conv.endian]
  SourceID              : [/mnt/ecloud/workspace/Build_FBO_OpenSys_r11.1.
                          1.0.11 001 [41228]/perforce/src/gglib/ggstd/
                          lecnv.c]
  SourceFunction        : [convCompSQL]
  SourceLine            : [531]
  ThreadBacktrace       : [9] elements
                        : [/opt/GoldenGate/orcl2/replicat(CMessageContext::
                          AddThreadContext()+0x26) [0x82021d6]]
                        : [/opt/GoldenGate/orcl2/replicat(CMessageFactory::
                          CreateMessage(CSourceContext*, unsigned int, ...)
                          +0x817) [0x81f8887]]
                        : [/opt/GoldenGate/orcl2/replicat( MSG ERR MAP
                          COL_INDEX_INVALID(CSourceContext*,  DBString<777>
                          const&, int, int, CMessageFactory::Message-
                          Disposition)+0x8b) [0x81d6c4b]]
                        : [/opt/GoldenGate/orcl2/replicat [0x84aa2bc]]
                        : [/opt/GoldenGate/orcl2/replicat(ggConvRecLE(char*,
                          file_def*, int, char, char)+0x4d) [0x84aa3bd]]
```

```
                          : [/opt/GoldenGate/orcl2/replicat [0x849dd2d]]
                          : [/opt/GoldenGate/orcl2/replicat(main+0x1f8b)
                            [0x812670b]]
                          : [/lib/libc.so.6(__libc_start_main+0xdc)
                            [0x68de8c]]
                          : [/opt/GoldenGate/orcl2/replicat(  gxx
                            personality v0+0x1b5) [0x810a171]]

2011-03-22 00:36:37  ERROR   OGG-01161  Bad column index (24144) specified
for table SCOTT.TEST, max columns = 2.
```

根据错误信息猜测是由于抽取进程加密了 trail 文件，Replicat 进程无法还原为真实的信息，导致了进程 abend。

下面在容灾端参数文件中加入 DECRYPTTRAIL 参数，让其对 trail 文件解密并查看进程的状态：

示例 10-6：

```
GGSCI (OE5) 3> view params repma

REPLICAT repma
USERID GoldenGate@orcl2, PASSWORD GoldenGate
setenv (NLS LANG="AMERICAN AMERICA.WE8ISO8859P1")
--REPORT AT 01:59
REPORTCOUNT EVERY 30 MINUTES, RATE
REPERROR DEFAULT, abend
DECRYPTTRAIL   -----加入解密参数
numfiles 50000
DBOPTIONS ALLOWUNUSEDCOLUMN
MAXTRANSOPS  500000
GROUPTRANSOPS 10000
CHECKPOINTSECS 40
--HANDLECOLLISIONS
assumetargetdefs
DISCARDFILE ./dirrpt/repma.dsc, APPEND, MEGABYTES 50
GETTRUNCATES
ALLOWNOOPUPDATES

map scott.* , target scott.* ;


GGSCI (OE5) 14> info all

Program     Status      Group       Lag           Time Since Chkpt

MANAGER
REPLICAT    RUNNING     REPMA       00:00:00      00:00:03
```

加入解密参数后重新启动 Replicat 进程，Replicat 进程显示 running 状态。

10.2　加密数据库密码

可以通过 GoldenGate 加密一些数据库口令，可以加密的数据库口令大致有下列 3 种。

- GoldenGate Extract、Replicat 进程及其他进程登录到数据库的密码。
- ASM 数据库、GoldenGate 需要登录到 ASM 实例的密码。
- GoldenGate 开启 DDL 的情况下，如果生产端执行类似 CREATE | ALTER} USER <name> IDENTIFIED BY <password> 的操作，容灾端有参数 DDLOPTIONS DEFAULTUSERPASSWORD 就会对密码进行加密，使其与生产端的不同。

加密数据库密码的方法如下。

进入 GoldenGate GGSCI 命令行，然后输入命令：

示例 10-7：

```
ENCRYPT PASSWORD <password>
```

GoldenGate 就会利用默认的 key 生成一个加密的密码，当然也可以自己指定 key 来生成加密密码，只需要键入命令：

示例 10-8：

```
CRYPT PASSWORD <password> ENCRYPTKEY <keyname>
```

<keyname>是用户自己生成的 KEY 的一个名字，这个名字和 KEY 将会保存在本地的 ENCKEYS 文件中。当然想使用这个属性，必须生成一个 KEY，而且在本地创建一个 ENCKEYS 文件，并且为这个 KEY 创建一个名字，那就是 keyname。

在用到 encryptkey 属性时候，有必要先介绍一下生成 encryption keys 的方法。

用户自己定义 KEY：首先要创建一个 1 到 24 个字符的 keyname，其中不能包含空格和引用，keyvalues 最大为 128 个字节，可以包含数字和字母或者是一个十六进制的字符串加上十六进制的标识符 0x，例如：0x420E61BE7002D63560929CCA17A4E1FB。

利用 KEYGEN 属性生成 KEY：源端在 GoldenGate 的安装目录下，在 shell 下键入命令：

示例 10-9：

```
KEYGEN <key length> <n>
```

可以得到多个 KEY，其中：

<key length>：是生成的加密密码的长度，最大为 128 字节。

<n>：控制要生成的 KEY 的数量。

示例 10-10：

```
[oracle@OE5 orcl1]$ ./keygen 128 4
```

```
0xA3116324F0C72B3BE328E728C6E75725

0x907B7678A7AB561CAF2532539A1DE72A

0x7EE5894C5D8F817D7B227D7D6E537630

0x6C4F9D201473AC5E481FC82742890536

[oracle@OE5 orcl1]$
```

创建一个名为 ENCKEYS 的 ASCII 文件，为生成的每个 KEY 起一个名字并保存到这个文件中，以便于 GoldenGate 使用：

示例 10-11：

```
## Encryption keys
## Key name        Key value
superkey          0xA3116324F0C72B3BE328E728C6E75725
superkey1         0x907B7678A7AB561CAF2532539A1DE72A
superkey2         0x7EE5894C5D8F817D7B227D7D6E537630
superkey3         0x6C4F9D201473AC5E481FC82742890536
```

然后，使用 GoldenGate 默认的 KEY 来加密数据库密码：

示例 10-12：

```
[oracle@OE5 orcl1]$ ./ggsci

Oracle GoldenGate Command Interpreter for Oracle
Version 11.1.1.0.11 Build 001
Linux, x86, 32bit (optimized), Oracle 10 on Dec  6 2010 14:20:28
Copyright (C) 1995, 2010, Oracle and/or its affiliates. All rights reserved.

GGSCI (OE5) 1> ENCRYPT PASSWORD GoldenGate
No key specified, using default key…

Encrypted password:  AACAAAAAAAAAAAKAPATACEHBIGQGCFZCCDIGAEMCQFFBZHVC
--这就是生成的加密密码

GGSCI (OE5) 2>
```

复制生成的加密密码按下列方式粘贴到 GoldenGate 参数文件中。

GoldenGate 用户密码：

示例 10-13：

```
USERID <user>, PASSWORD <encrypted password>, &ENCRYPTKEY {DEFAULT |
<keyname>}
GGSCI (OE5) 5> edit params extma

EXTRACT extma
--userid GoldenGate@orcl1, password GoldenGate
```

```
userid GoldenGate@orcl1 , password AACAAAAAAAAAAAKAPATACEHBIGQGCFZCCDIGA-
EMCQFFBZHVC , ENCRYPTKEY DEFAULT
setenv (NLS LANG="AMERICAN AMERICA.WE8ISO8859P1")
GETTRUNCATES
REPORTCOUNT EVERY 1 MINUTES, RATE
numfiles 50000
DISCARDFILE ./dirrpt/extma.dsc,APPEND,MEGABYTES 50
WARNLONGTRANS 2h,CHECKINTERVAL 3m
EXTTRAIL ./dirdat/ma
DBOPTIONS ALLOWUNUSEDCOLUMN
TRANLOGOPTIONS  CONVERTUCS2CLOBS
DYNAMICRESOLUTION
table scott.* ;
```

这样在打开参数文件的时候，就看不到密码的明文了。黑客即使攻破了 GoldenGate 用户，看到这个配置文件，用里面这个加密的密码也无法登录到数据库，这样就起到了保护数据库数据的作用。

ASM GoldenGate 用户访问密码：

示例 10-14：

```
TRANLOGOPTIONS ASMUSER SYS@<ASM instance name>, ASMPASSWORD <encrypted
password>, ENCRYPTKEY {DEFAULT | <keyname>}
```

读者可以自行去试验，这里就不演示了。

CREATE/ALTER USER 密码：

示例 10-15：

```
DDLOPTIONS DEFAULTUSERPASSWORD <encrypted password>,ENCRYPTKEY {DEFAULT |
<keyname>}
```

对参数中名词的解释：

- **<user id>**　是数据库中用于 GoldenGate 进程的用户。对于 ASM，用户必须具有 SYS 权限。
- **<encrypted_password>**　使用命令 ENCRYPT PASSWORD 得出的加密密码。
- **ENCRYPTKEY DEFAULT**　利用 GoldenGate 默认的 KEY 生成的加密密码。
- **ENCRYPTKEY <keyname>**　如果在使用命令 ENCRYPT PASSWORD 的时候使用了 ENCRYPTKEY <keyname>参数，那么在参数文件中也需要加入这个选项。告诉 GoldenGate 是使用用户自定义的 KEY 生成的加密密码。

10.3　网络传输加密

GoldenGate 在传输数据的时候，默认是不加密的。可以在 GoldenGate 通过网络传输数据之前将数据加密，传送到目标端以后，在写入 trail 文件之前将数据解密。这样就有效地

保护了数据在传输过程中的安全。

加密网络传输的步骤如下。

在生产端生成多个 KEY，然后保存到 ENCKEYS 文件中，上面介绍生成了 ENCKEYS 文件，这里我们直接拿来用。

示例 10-16：

```
[oracle@OE5 orcl1]$ cat ENCKEYS
## Encryption keys
## Key name
superkey          0xA3116324F0C72B3BE328E728C6E75725
superkey1         0x907B7678A7AB561CAF2532539A1DE72A
superkey2         0x7EE5894C5D8F817D7B227D7D6E537630
superkey3         0x6C4F9D201473AC5E481FC82742890536
```

把这个文件 copy 到容灾端的 GG 安装目录下。

容灾端的 ENCKEYS 文件中的内容必须和生产端的完全一样，不然 GG 会报错：

示例 10-17：

```
2011-03-22 21:18:59  ERROR   OGG-01224  TCP/IP error 104 (Connection reset
by peer); retries exceeded.

2011-03-22 21:18:59  ERROR   OGG-01668  PROCESS ABENDING.
```

使用 ENCRYPT 选项的 RMTHOST 参数来指定加密的类型和要使用的在 ENCKEYS 文件中列出的 KEY：

示例 10-18：

```
ENCRYPT BLOWFISH, KEYNAME <keyname>
```

其中，BLOWFISH 是要使用的算法，这种算法加密对性能的损失相对较小。

<keyname>是 ENCKEYS 文件中列出的 keyname。

示例 10-19：

```
GGSCI (OE5) 44> view params dpema

EXTRACT dpema
RMTHOST 192.168.50.200, MGRPORT 7849, ENCRYPT BLOWFISH, KEYNAME superkey
--RMTHOST 192.168.50.200 , MGRPORT 7849, compress
PASSTHRU
numfiles 50000
RMTTRAIL ./dirdat/ma
DYNAMICRESOLUTION

table scott.* ;
```

查看容灾端 MGR 进程 report，可以看到“encrypt BLOWFISH -keyname SUPERKEY”，

表明 GG 已经实现了加密处理。

示例 10-20：

```
2011-03-22 21:12:03  INFO    OGG-00963  Command received from EXTRACT on
host 192.168.50.200 (START SERVER CPU -1 PRI -1  TIMEOUT 300 PARAMS  -encrypt
BLOWFISH -keyname SUPERKEY).
```

10.4 使用 cmdsec 进行权限控制

GoldenGate 可以限制一些用户对命令的使用。例如可以让一些监控用户只使用 INFO 和 STAT 命令，不允许其使用 start 和 stop 命令，这是通过对操作系统用户组的限制而是现在限制使用命令的目的。

GoldenGate 使用通过在安装目录下创建一个文件，并在文件加入一些规则来控制用户对命令的使用，每一行一条规则，规则的顺序要从控制范围最广到范围最少从上到下写，每个规则要用空格隔开。书写规则的格式如下：

示例 10-21：

```
<command name> <command object> <OS group> <OS user> <YES | NO>
```

对每个选项的解释：

<command name>：是 GoldenGate 的命令或者一个通配符，例如 start、stop、*。

<command object>：是 GoldenGate 的进程组类型或通配符，例如 Extract、Replicat、MGR。

<OS group>：是操作系统用户组，在 UNIX 下可以用户 ID 来代替用户名，或者使用*来表示所有的用户组。

<YES | NO>：表示这个命令对这个用户是开放的还是禁止的。

下面是一个 Linux 系统下控制用户对命令使用的一个简单的案例：

示例 10-22：

```
#GG command security  命令行
STATUS REPLICAT * Smith NO  --不允许 smith 在容灾端使用 STATUS 命令
STATUS * dpt1 * YES         --除了上面的规则，所有 dpt1 组下的用户可以使用 status
                              命令
START REPLICAT root * YES   --root 组下的用户可以使用 start Replicat 命令
START REPLICAT * * NO       --除了上面的规则，所有用户不能使用 start replicat 命令
* EXTRACT 200 * NO          --group id 为 200 的组在生产端不可以使用命令
* * root root YES           --root 用户可以使用任何的命令
* * * * NO                  --除了上面的规则，所有的用户不可以使用 GoldenGate 命令
```

CMDSEC 文件是命令行安全的根源，必须要保证这个文件的安全，可以赋予用户对它读的权限，但是不允许 GoldenGate 管理员以外的所有用户修改和删除这个文件。

第 11 章　对 GoldenGate 的监控

对 GoldenGate 实例进行监控，最简单高效的办法是通过 GGSCI 命令行的方式进行。通过在命令行输入一系列命令，并查看返回信息，来判断 GoldenGate 运行情况是否正常。命令行返回的信息包括整体概况、进程运行状态、检查点信息、参数文件配置、延时等。

11.1　使用 GGSCI 命令监控

（1）进入 GoldenGate 安装目录，运行 GGSCI，然后使用 info all 查看整体的运行状况，如图 11-1 所示。

```
GGSCI (node1) 12> info all

Program     Status      Group       Lag           Time Since Chkpt

MANAGER     RUNNING
EXTRACT     RUNNING     EORA        00:00:00      00:00:05
EXTRACT     RUNNING     PUMP_SO     00:00:00      00:00:07
```

图 11-1

Group 表示进程的名称（MGR 进程不显示名字）；Lag 表示进程的延时；Status 表示进程的状态，有 4 种状态。

- **STARTING**　表示正在启动过程中。
- **RUNNING**　表示进程正常运行。
- **STOPPED**　表示进程被正常关闭。
- **ABENDED**　表示进程非正常关闭，需要进一步调查原因。

正常情况下，所有进程的状态应该为 RUNNING，且 Lag 应该在一个合理的范围内。

（2）使用 view params <进程名> 可以查看进程的参数设置，该命令支持通配符*，如图 11-2 所示。

```
GGSCI (node1) 14> view params mgr

port 7809
dynamicportlist 7800-8000
autorestart extract *,waitminutes 2,resetminutes 5
```

图 11-2

（3）使用 info <进程名称> 命令可以查看进程信息，可以查看到的信息包括进程状态、checkpoint 信息、延时等，如图 11-3 所示。

```
GGSCI (node1) 15> info eora

EXTRACT    EORA      Last Started 2011-01-08 00:50   Status RUNNING
Checkpoint Lag       00:00:00 (updated 00:00:09 ago)
Log Read Checkpoint  Oracle Redo Logs
                     2011-01-08 01:04:36  Seqno 9, RBA 958976
```

图 11-3

（4）还可以使用 info <进程名称> detail 命令查看更详细的信息，包括所使用的 trail 文件、参数文件、报告文件、警告日志的位置等，如图 11-4 和图 11-5 所示。

```
GGSCI (node1) 16> info eora detail

EXTRACT    EORA      Last Started 2011-01-08 00:50   Status RUNNING
Checkpoint Lag       00:00:00 (updated 00:00:07 ago)
Log Read Checkpoint  Oracle Redo Logs
                     2011-01-08 01:04:36  Seqno 9, RBA 958976

  Target Extract Trails:

  Remote File Name                                   RBA     Max MB

  /u01/ggs/10.0/dirdat/et                              8       1092         10

  Extract Source                        Begin             End

  /u01/app/oracle/oradata/orcl/redo02.log  2010-12-18 03:31  2011-01-08 01:04
```

图 11-4

```
Current directory    /u01/ggs/10.0

Report file          /u01/ggs/10.0/dirrpt/EORA.rpt
Parameter file       /u01/ggs/10.0/dirprm/eora.prm
Checkpoint file      /u01/ggs/10.0/dirchk/EORA.cpe
Process file         /u01/ggs/10.0/dirpcs/EORA.pce
Stdout file          /u01/ggs/10.0/dirout/EORA.out
Error log            /u01/ggs/10.0/ggserr.log
```

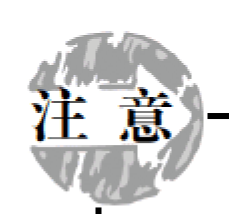
图 11-5

（5）使用 info <进程名称> showch 命令可以查看到详细的关于 checkpoint 的信息，用于查看 GoldenGate 进程处理过的事务记录，如图 11-6 所示。

其中比较重要的是 Extract 进程的 recovery checkpoint，它表示源数据中最早的未被处理的事务；通过 recovery checkpoint 可以查看到该事务的 redo log 位于哪个日志文件以及该日志文件的序列号。

所有序列号比它大的日志文件，均需要保留。

（6）lag <进程名称> 可以查看详细的延时信息，如图 11-7 所示。



此命令只能够查看到最后一条处理过的记录的延时信息。

```
GGSCI (node1) 18> info eora showch

EXTRACT    EORA      Last Started 2011-01-08 00:50   Status RUNNING
Checkpoint Lag       00:00:00 (updated 00:00:02 ago)
Log Read Checkpoint  Oracle Redo Logs
                     2011-01-08 01:09:00  Seqno 9, RBA 969728


Current Checkpoint Detail:

Read Checkpoint #1

  Oracle Redo Log

  Startup Checkpoint (starting position in the data source):
    Sequence #: 8
    RBA: 24282640
    Timestamp: 2010-12-18 03:31:34.000000
    Redo File: /u01/app/oracle/oradata/orcl/redo01.log

  Recovery Checkpoint (position of oldest unprocessed transaction in the
    Sequence #: 9
    RBA: 968720
    Timestamp: 2011-01-08 01:09:00.000000
    Redo File: /u01/app/oracle/oradata/orcl/redo02.log

  Current Checkpoint (position of last record read in the data source):
    Sequence #: 9
    RBA: 969728
    Timestamp: 2011-01-08 01:09:00.000000
    Redo File: /u01/app/oracle/oradata/orcl/redo02.log
```

图 11-6

```
GGSCI (node1) 20> lag eora

Sending GETLAG request to EXTRACT EORA ...
Last record lag: 1 seconds.
At EOF, no more records to process.
```

图 11-7

（7）stats 可以查看进程处理的记录数：

示例 11-1：

```
stats <进程名称>,<时间频度>,table <owner name>.<table name>
```

该报告会详细地列出处理的类型和记录数，例如：

示例 11-2：

```
GGSCI> stats eora, total
```

列出自进程启动以来处理的所有记录数，如图 11-8 所示。

示例 11-3：

```
GGSCI> stats eora, daily, table scott.dept
```

```
GGSCI (node1) 23> stats eora total

Sending STATS request to EXTRACT EORA ...

Start of Statistics at 2011-01-08 01:15:30.

Output to /u01/ggs/10.0/dirdat/et:

Extracting from SCOTT.DEPT to SCOTT.DEPT:

*** Total statistics since 2011-01-08 00:54:03 ***
        Total inserts                              0.00
        Total updates                              0.00
        Total deletes                              1.00
        Total discards                             0.00
        Total operations                           1.00

End of Statistics.
```

图 11-8

列出当天以来处理的有关 scott.dept 表的所有记录数。

（8）view report <进程名称> 可以查看运行报告，如图 11-9 所示。

```
GGSCI (node1) 26> view report eora


***********************************************************************
                Oracle GoldenGate Capture for Oracle
                     Version 10.4.0.19 Build 002
   Linux, x86, 32bit (optimized), Oracle 10 on Sep 25 2009 12:56:40

Copyright (C) 1995, 2009, Oracle and/or its affiliates.  All rights reserved.


                    Starting at 2011-01-08 00:50:23
***********************************************************************

Operating System Version:
Linux
```

图 11-9

（9）也可以进入到 <GoldenGate 安装目录>/dirrpt/目录下，查看对应的报告文件。

最新的报告总是以<进程名称>.rpt 命名的。加后缀数字的报告是历史报告，数字越大对应的时间越久，如图 11-10 所示。

```
[oracle@node1 dirrpt]$ ls
EORA0.rpt  EORA5.rpt  EORA.rpt  MGR.rpt       PUMP_SO4.rpt  PUMP_SO.rpt
EORA1.rpt  EORA6.rpt  MGR0.rpt  PUMP_SO0.rpt  PUMP_SO5.rpt
EORA2.rpt  EORA7.rpt  MGR1.rpt  PUMP_SO1.rpt  PUMP_SO6.rpt
EORA3.rpt  EORA8.rpt  MGR2.rpt  PUMP_SO2.rpt  PUMP_SO7.rpt
EORA4.rpt  EORA9.rpt  MGR3.rpt  PUMP_SO3.rpt  PUMP_SO8.rpt
```

图 11-10

如果进程运行时有错误，则报告文件中会包括错误代码和详细的错误诊断信息。通过

查找错误代码，可以帮助定位错误原因，解决问题。

11.2 ggserr.log 日志监控

（1）可以用以下方法查看日志文件。

- 可以通过操作系统命令直接查看 ggserr.log 文件。
- 使用 GoldenGate Director。
- 在 GGSCI 中运行命令 view ggsevt。

（2）在日志文件中可以查看到的内容如下。

- GGSCI 命令的历史记录。
- GoldenGate 进程的启动与停止。
- 已执行的处理。
- 发生的错误。
- 信息和警告消息。

（3）日志文件的部分内容，如图 11-11 所示。

```
GGSCI (node1) 2> view ggsevt

12:54:45..INFO    399 ..: GGSCI command (oracle): edit params mgr.
12:55:58..INFO    399 ..: GGSCI command (oracle): start mgr.
12:55:58..INFO    330 ..: Manager started (port 7809).
12:59:30..INFO    399 ..: GGSCI command (oracle): add extract ext tranlog,begin now.
13:33:54..INFO    399 ..: GGSCI command (oracle): start mgr.
13:34:02..INFO    399 ..: GGSCI command (oracle): stop mgr.
13:34:03..INFO    301 ..: Command received from GGSCI on host 127.0.0.1 (STOP).
13:34:03..WARNING 331 ..: Manager is stopping at user request.
13:34:14..INFO    399 ..: GGSCI command (oracle): start mgr.
13:34:14..INFO    330 ..: Manager started (port 7809).
```

图 11-11

11.3 日常运维监控的自动化脚本

GoldenGate 自动化监控的自动脚本主要体现在怎么保证 GoldenGate 各个进程状态是什么，这个最好由监控软件来做，但是 GoldenGate 当前以及需要的归档号码，还是由 GoldenGate 必须用脚本来实现。

以下内容作为参考：

示例 11-4：

```
#!/usr/bin/ksh
export ORACLE_BASE=/oracle/
export ORACLE_SID=epmln1
export ORACLE_HOSTNAME=pmlnpdb1
export ORA_NLS33=/oracle/db/ocommon/nls/admin/data
```

```
export ORA CRS HOME=/oracle/crs
export ORACLE HOME=/oracle/db

export gglog=/GoldenGate/dirrpt/`date "+gg_monitor_%Y-%m-%d_%H:%M:%S"`


echo "############ gg process status #############">>$gglog
ps -ef|grep ext|grep -v grep>>$gglog
ps -ef|grep mgr|grep -v grep>>$gglog
echo "####################################">>$gglog
echo "############ gg trail file system usage #############">>$gglog
bdf /GoldenGate >>$gglog

echo "####################################">>$gglog
cd /GoldenGate
echo "############ info all ############">>$gglog
echo "info all"|./ggsci >>$gglog

echo "############ lag * ############">>$gglog
echo "lag *"|./ggsci >>$gglog

echo "####################################">>$gglog
echo "############ info er * ############">>$gglog
echo "info er *"|./ggsci >>$gglog

echo "####################################">>$gglog
echo "############ info er *,detail ############">>$gglog
echo "info er *,detail"|./ggsci >>$gglog

echo "####################################">>$gglog
echo "############ info er *,showch ############">>$gglog
echo "info er *,showch"|./ggsci >>$gglog

echo "####################################">>$gglog
echo "############ view params  mgr ############">>$gglog
echo "view params  mgr"|./ggsci >>$gglog

echo "####################################">>$gglog
echo "############ view params  EXTYA ############">>$gglog
echo "view params  EXTYA"|./ggsci >>$gglog

echo "####################################">>$gglog
echo "############ view params  EXTYB ############">>$gglog
echo "view params  EXTYB"|./ggsci >>$gglog

echo "####################################">>$gglog
echo "############ view params  DPEYA ############">>$gglog
```

```
echo "view params  DPEYA"|./ggsci >>$gglog

echo "####################################">>$gglog
echo "############ view params  DPEYB ############">>$gglog
echo "view params  DPEYB"|./ggsci >>$gglog

echo "####################################">>$gglog
echo "############ stats * ############">>$gglog
echo "stats *"|./ggsci >>$gglog

echo "####################################">>$gglog
echo "############ tail -1000f /GoldenGate/ggserr.log ############">>$gglog
tail -1000 /GoldenGate/ggserr.log>>$gglog

echo "####################################">>$gglog
echo "############ info er * ,showtrans ############">>$gglog
echo "send er * ,showtrans"|./ggsci >>$gglog
```

11.4　使用 GoldenGate Director 监控

11.4.1　GoldenGate Director 技术框架

GoldenGate 提供图形化的界面，GoldenGate Director 是一种多层的、客户端—服务器模型的应用，GoldenGate Director 使用可以从远程客户端来配置和管理 GoldenGate instance。

11.4.2　GoldenGate Director 组件

GoldenGate Director 主要由如图 11-12 所示的组件组成。

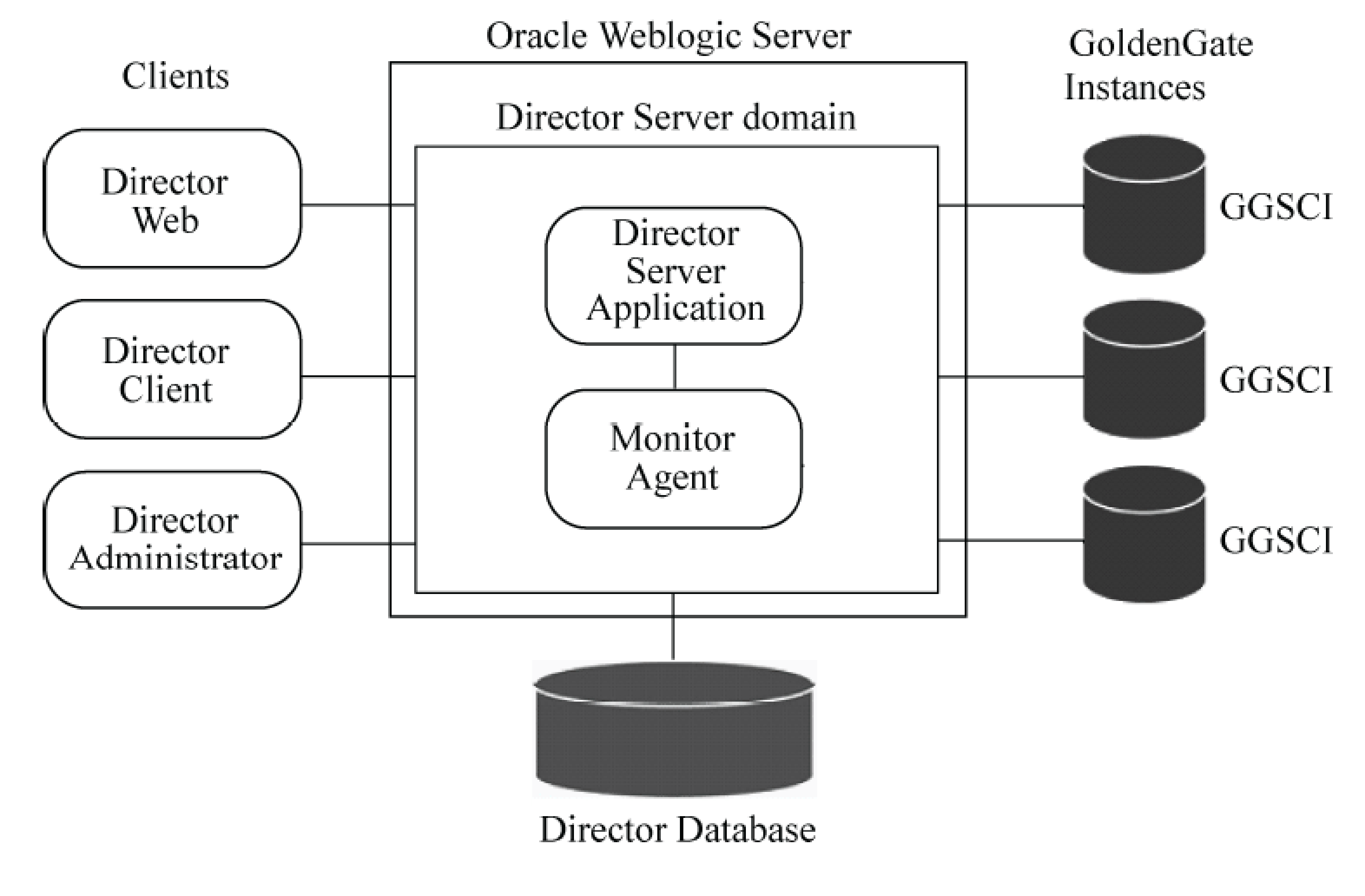


图 11-12

图 11-12 中：

（1）GoldenGate Instances 是需要被管理的实例。多个实例注册到同一个 GoldenGate Server 上，统一被管理。

（2）Director Server 是 Director 服务端，需要部署在 Weblogic Server 下。

（3）Director Database 用来存放 Director Server 资料库的数据库。

（4）Director Administrator 是管理 Director Server 的客户端工具，主要负责实例的注册和配置，必须和 Director Client 一起安装。

（5）Director Client 是一款客户端工具，也可以用于监控实例。必需安装此工具才能使用 Director Administrator。

（6）Director Web 是一款基于浏览器的监控工具，可以通过统一的 Web 界面监控多个 GoldeGate 实例。

11.4.3 GoldenGate Director 安装

在安装 GoldenGate Director 之前，需要先安装 JRE 6.0 版本和 Oracle WebLogic Server 11G 标准版（不需要预先设置 domain name）。另外需要数据库来存储 GoldenGate Director 的信息，可以使用 Oracle、SQL Server 以及 MySQL，本书选用 Oracle 数据库，并且创建好 GoldenGate Director 用户并赋予足够的权限（示例使用 dba）。

1. 安装 Oracle GoldenGate Director Server 端

这里是以 Windows 服务器为例：

（1）单击安装文件会出现安装的欢迎界面，如图 11-13 所示。

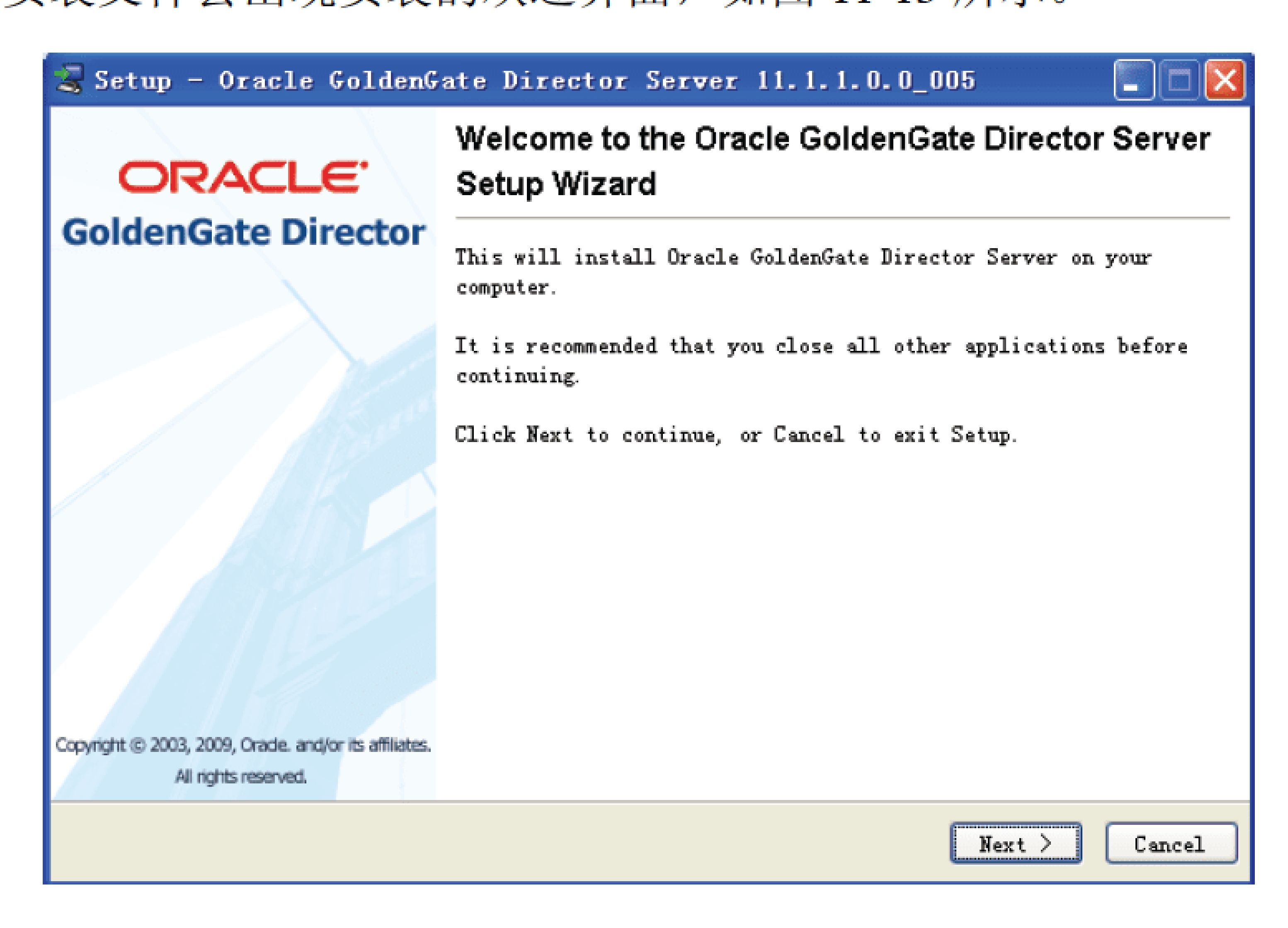


图 11-13

（2）单击 Next 按钮，选择 GG Director 的安装路径，如图 11-14 所示。

（3）选择默认路径，也可以修改为用户自己的路径，单击 Next 按钮，如图 11-15 所示。

（4）选择 WebLogic 所在的路径，选择好以后单击 Next 按钮，如图 11-16 所示。

（5）选择 HTTP 端口，如图 11-16 所示。

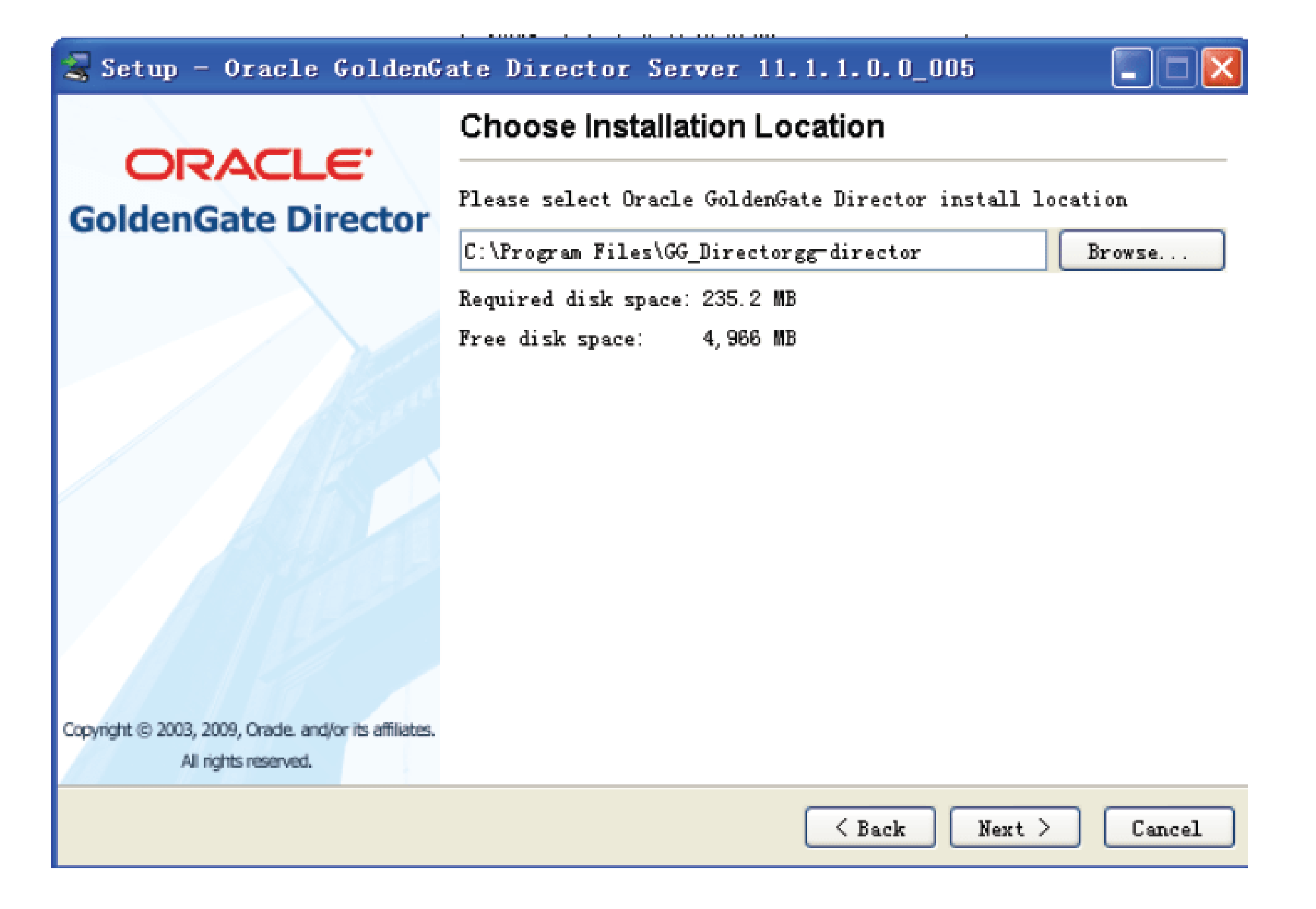


图 11-14

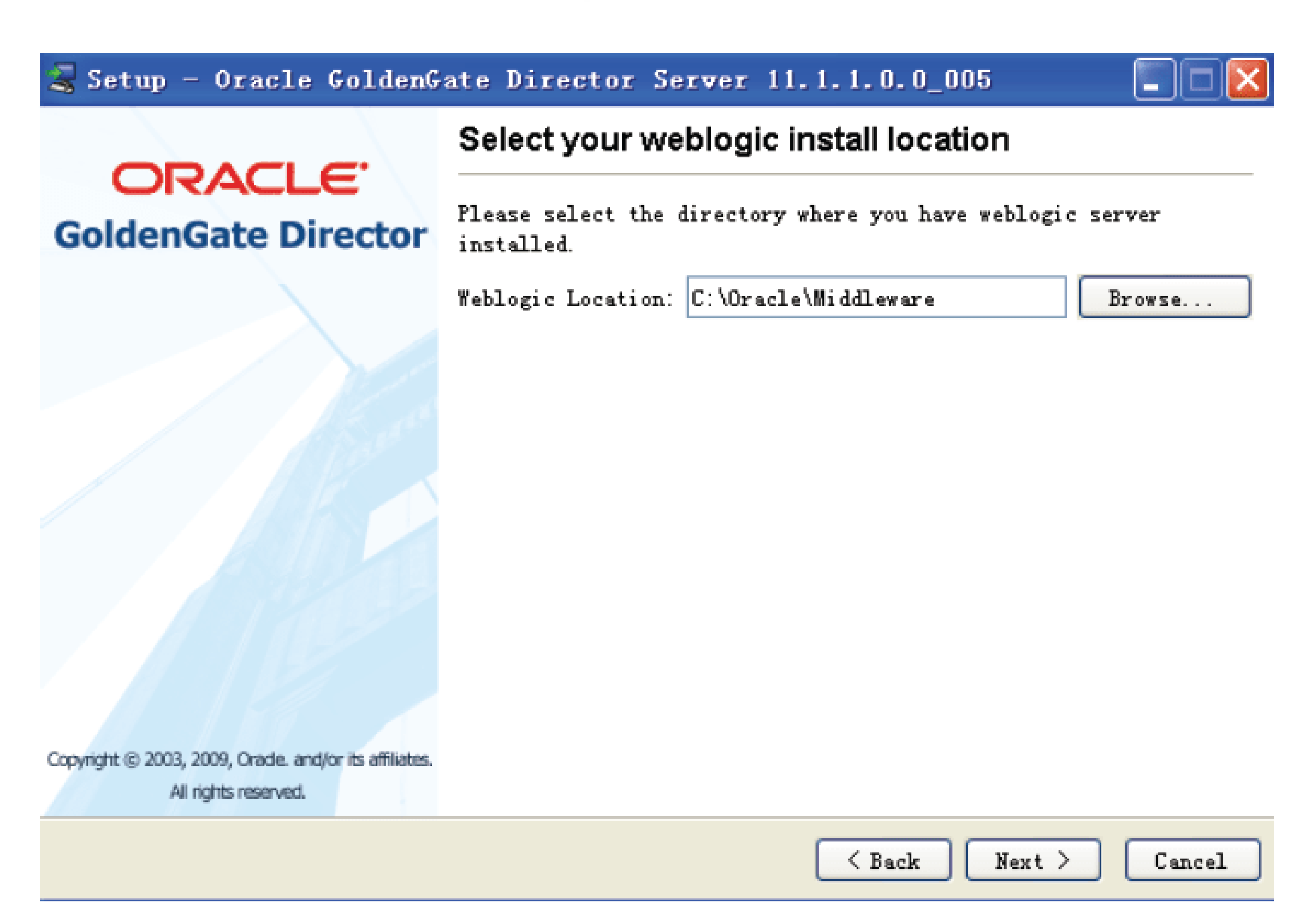


图 11-15

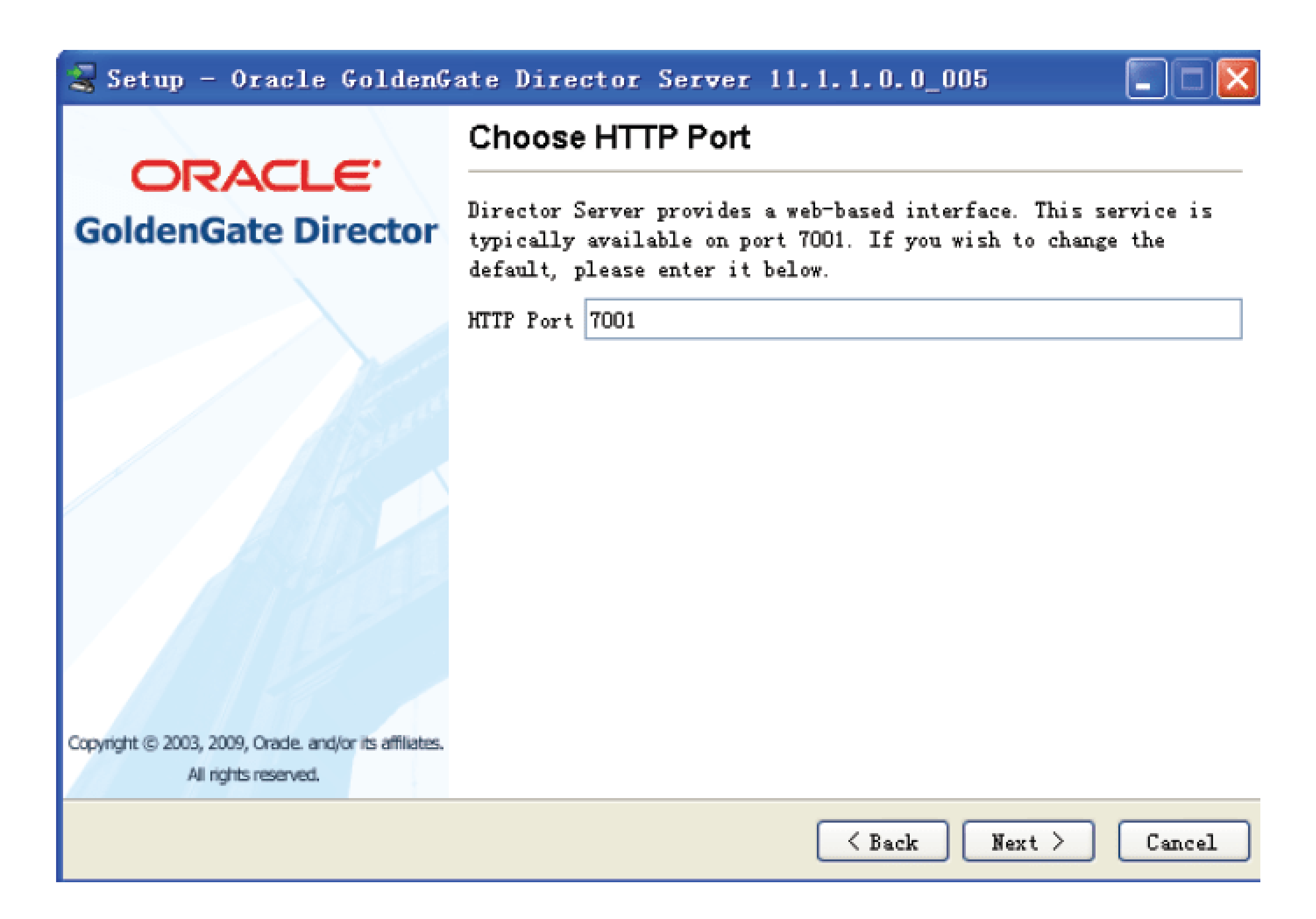


图 11-16

（6）选择一个存放 GoldenGate Director 信息的数据库（示例选择 Oracle），如图 11-17 所示。

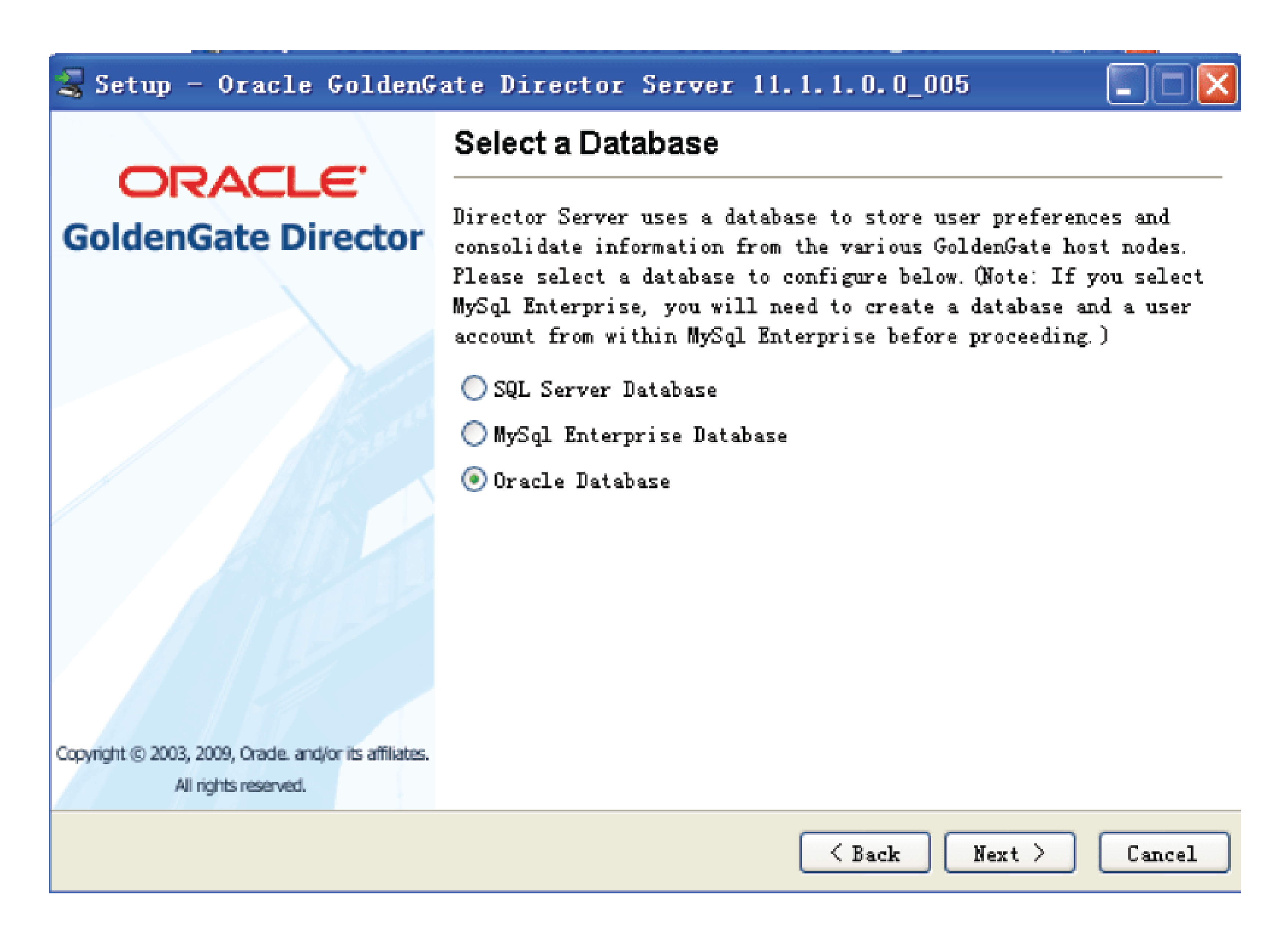


图 11-17

（7）配置数据库连接，如图 11-18 所示。

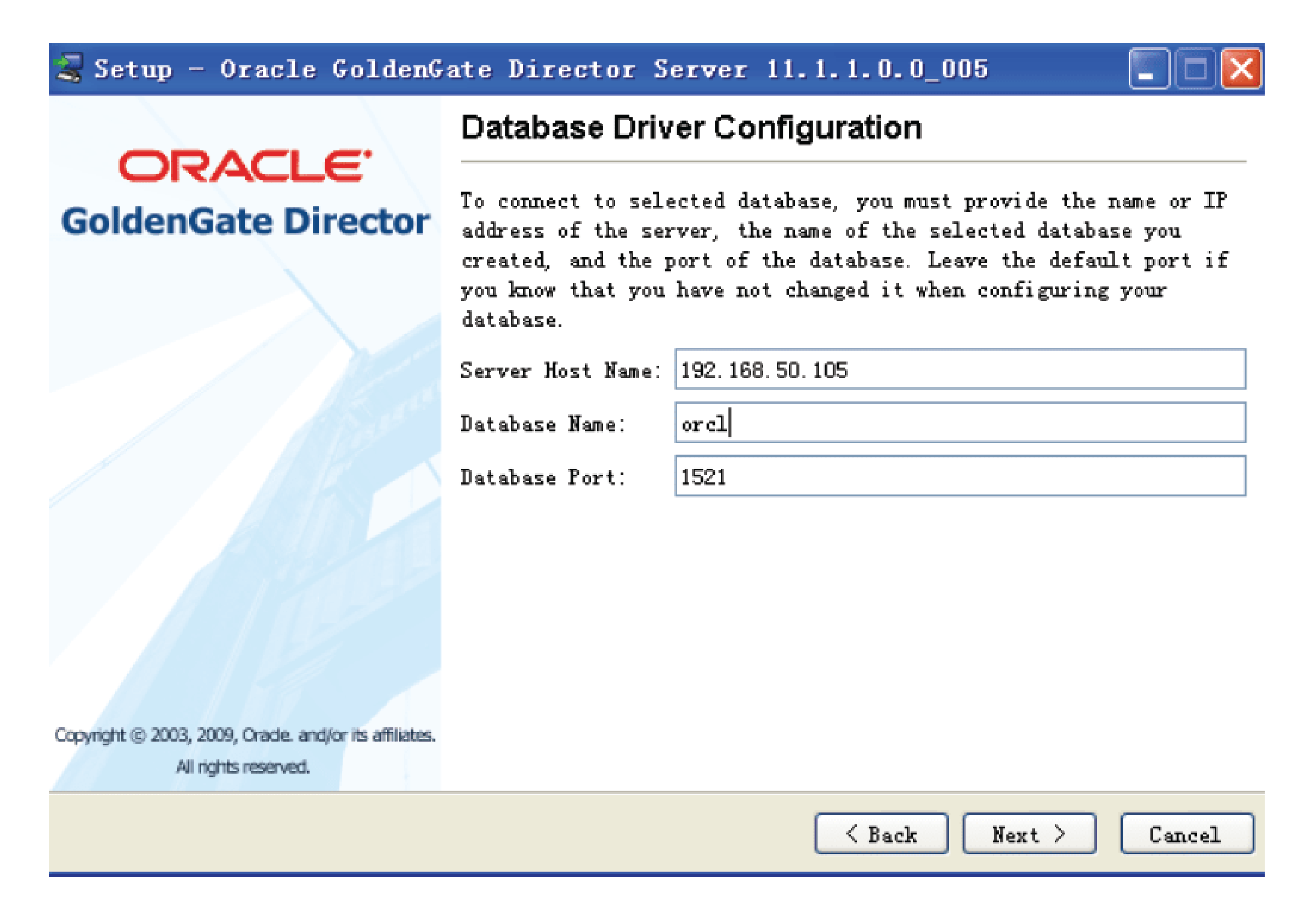


图 11-18

（8）数据存放 GoldenGate Director 信息的 Oracle 数据库的用户名及密码，如图 11-19 所示。

（9）确认安装的一些信息如图 11-20 所示，单击 Next 按钮，如图 11-21 所示。

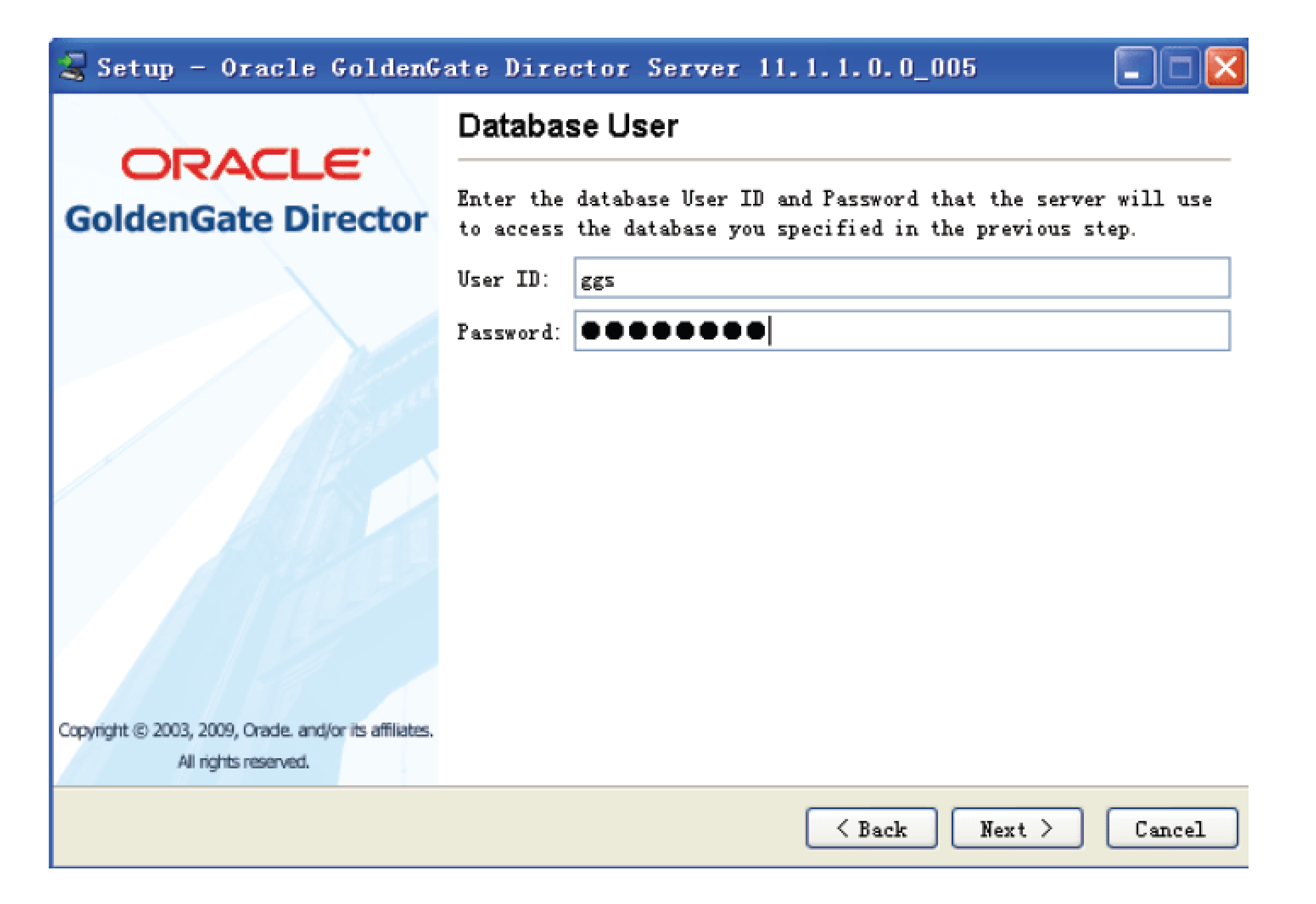


图 11-19

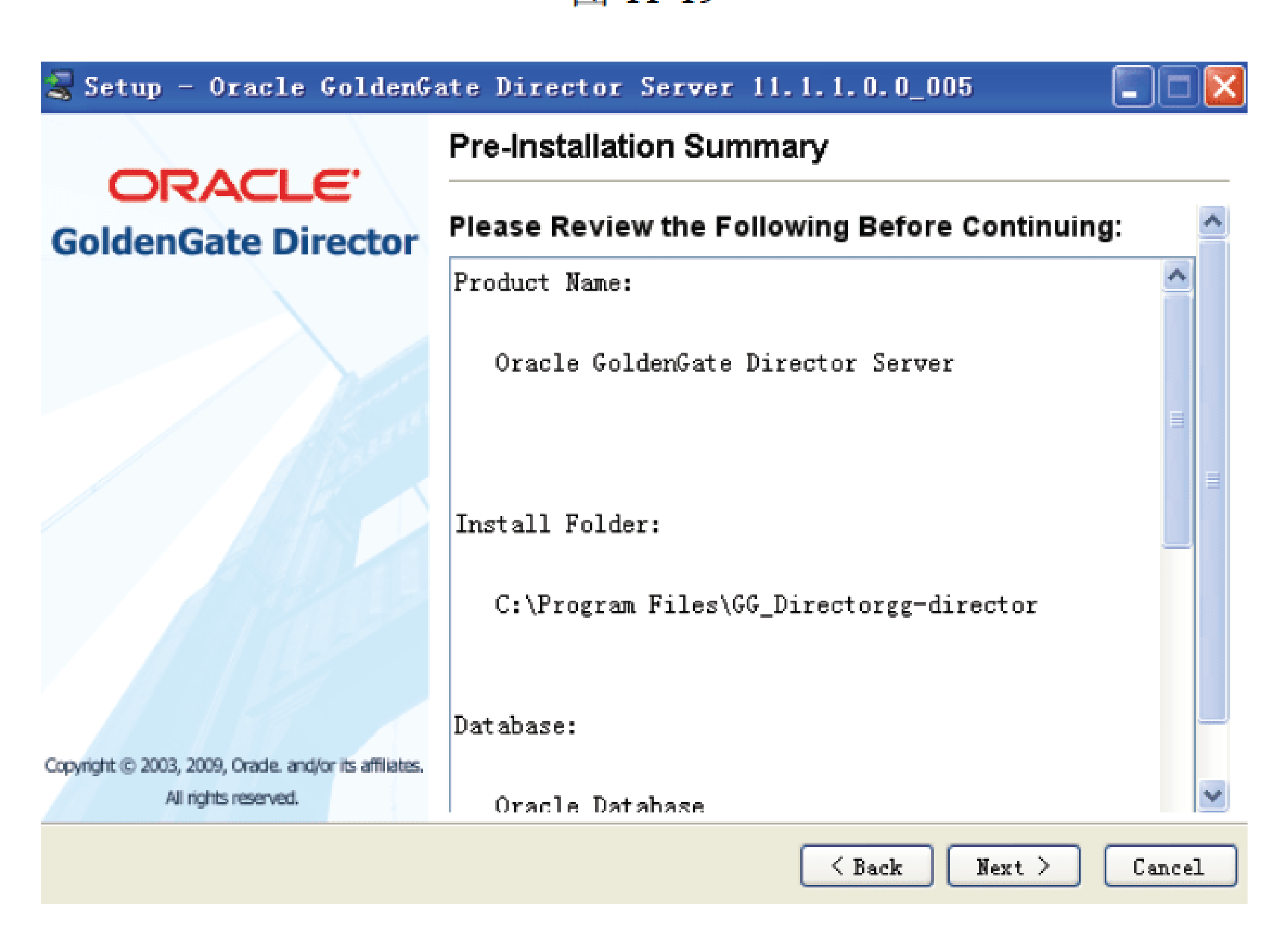


图 11-20



图 11-21

安装完成，如图 11-22 所示单击 Finish 按钮。

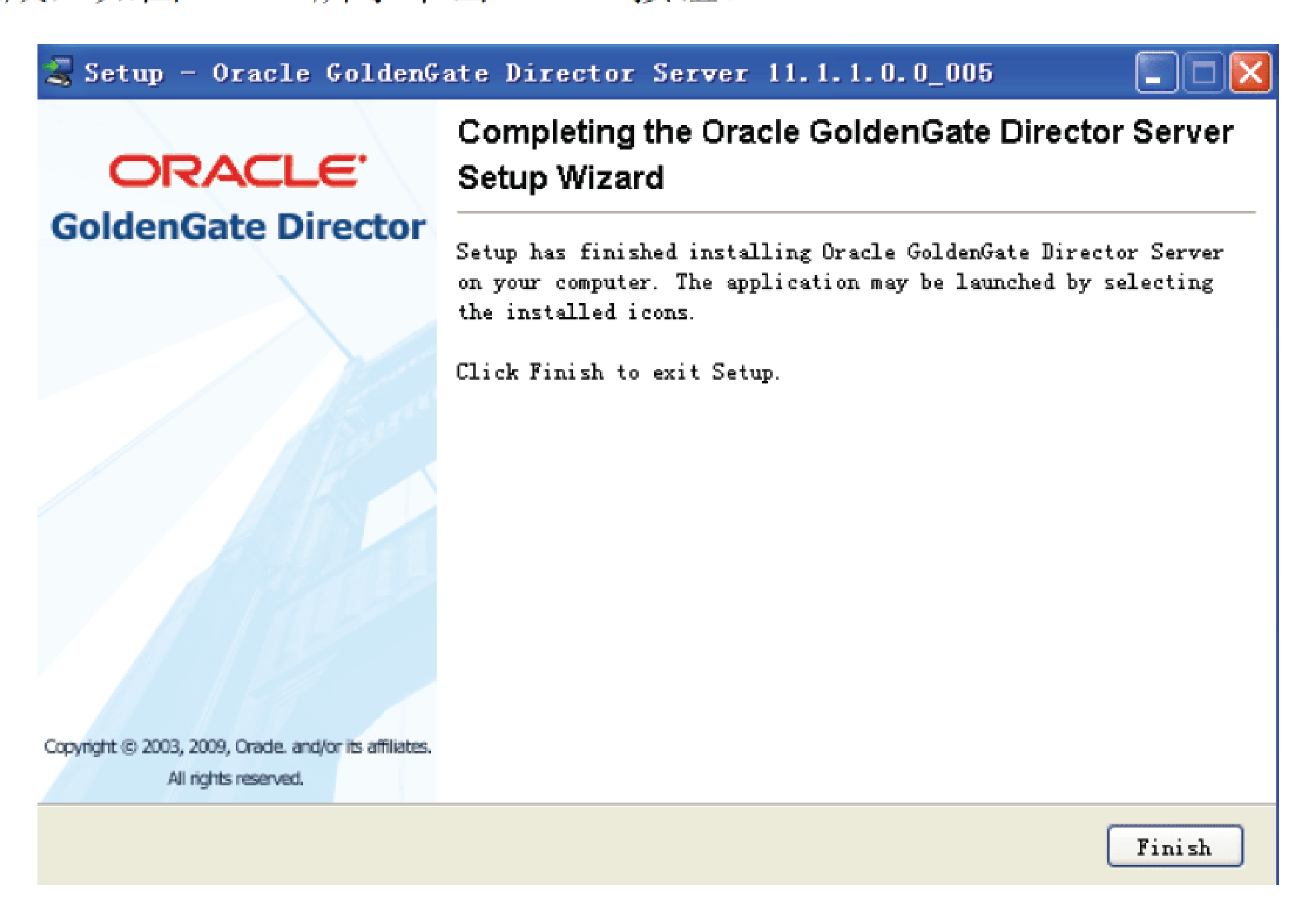


图 11-22

2．安装 Oracle GoldenGate Director 客户端

（1）单击安装文件，出现欢迎界面，如图 11-23 所示。

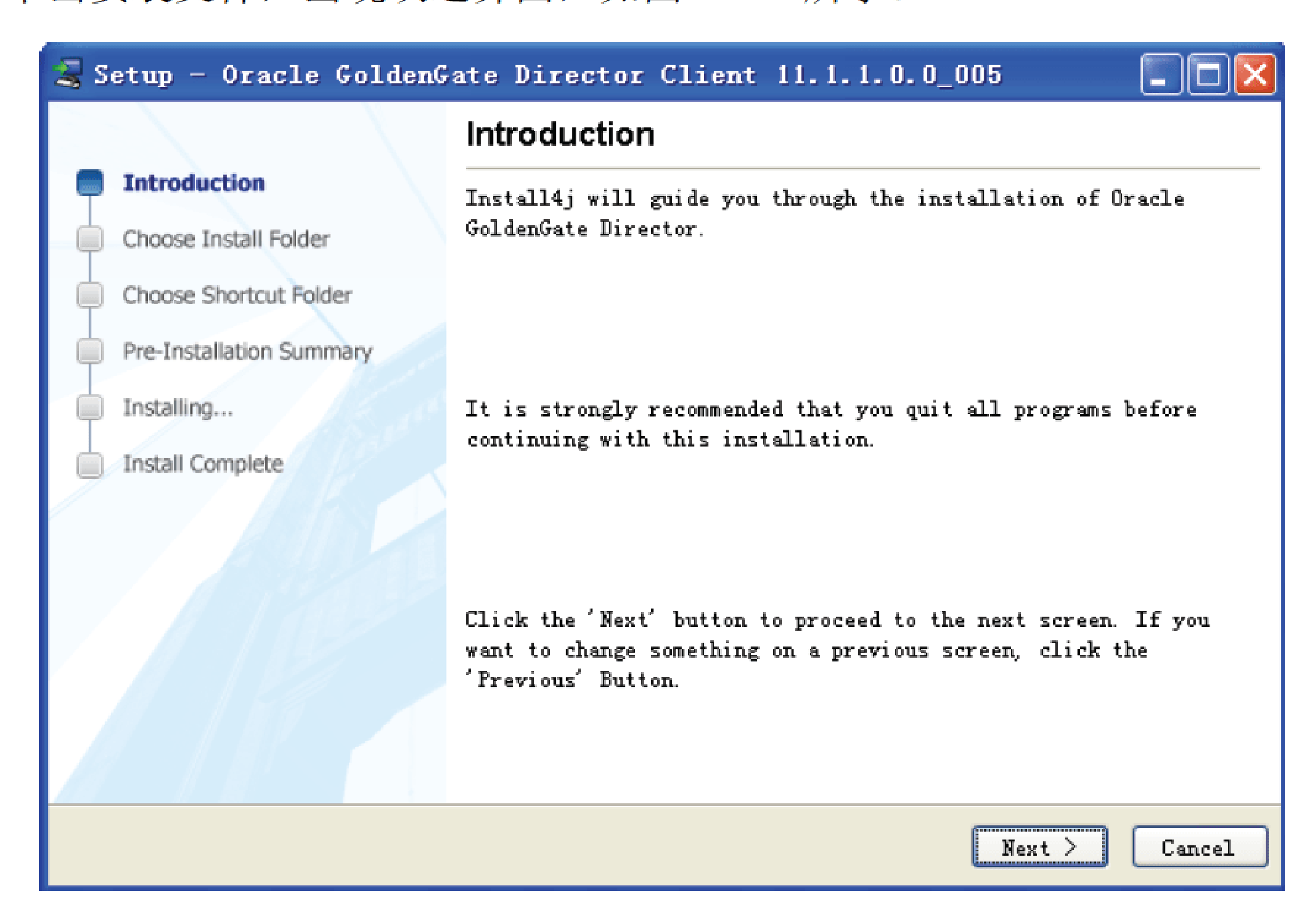


图 11-23

（2）单击 Next 按钮，如图 11-24 所示。

（3）选择安装路径，单击 Next 按钮，如图 11-25 所示。

（4）安装开始菜单选项。

（5）安装信息预览，如图 11-26 所示。

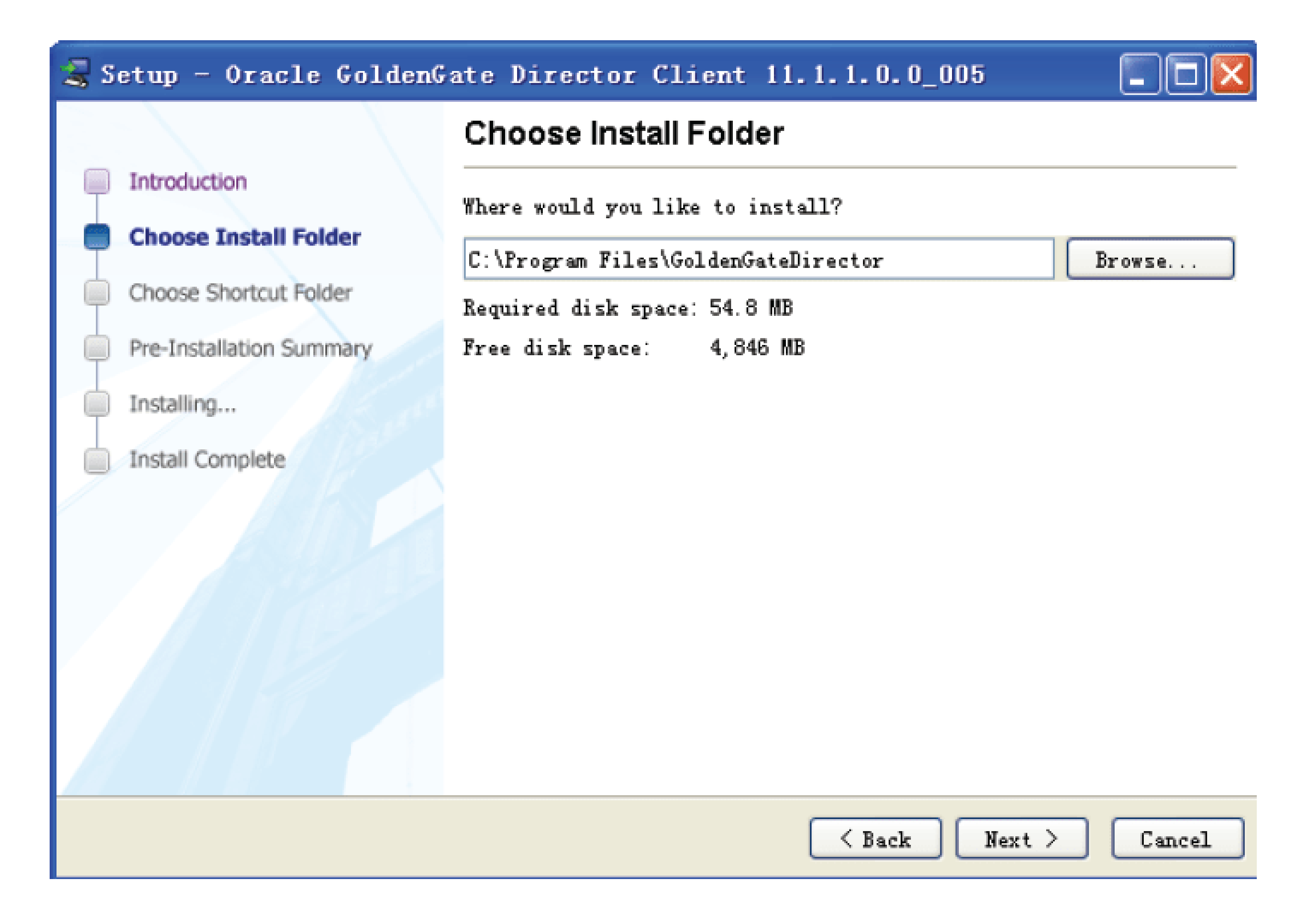


图 11-24

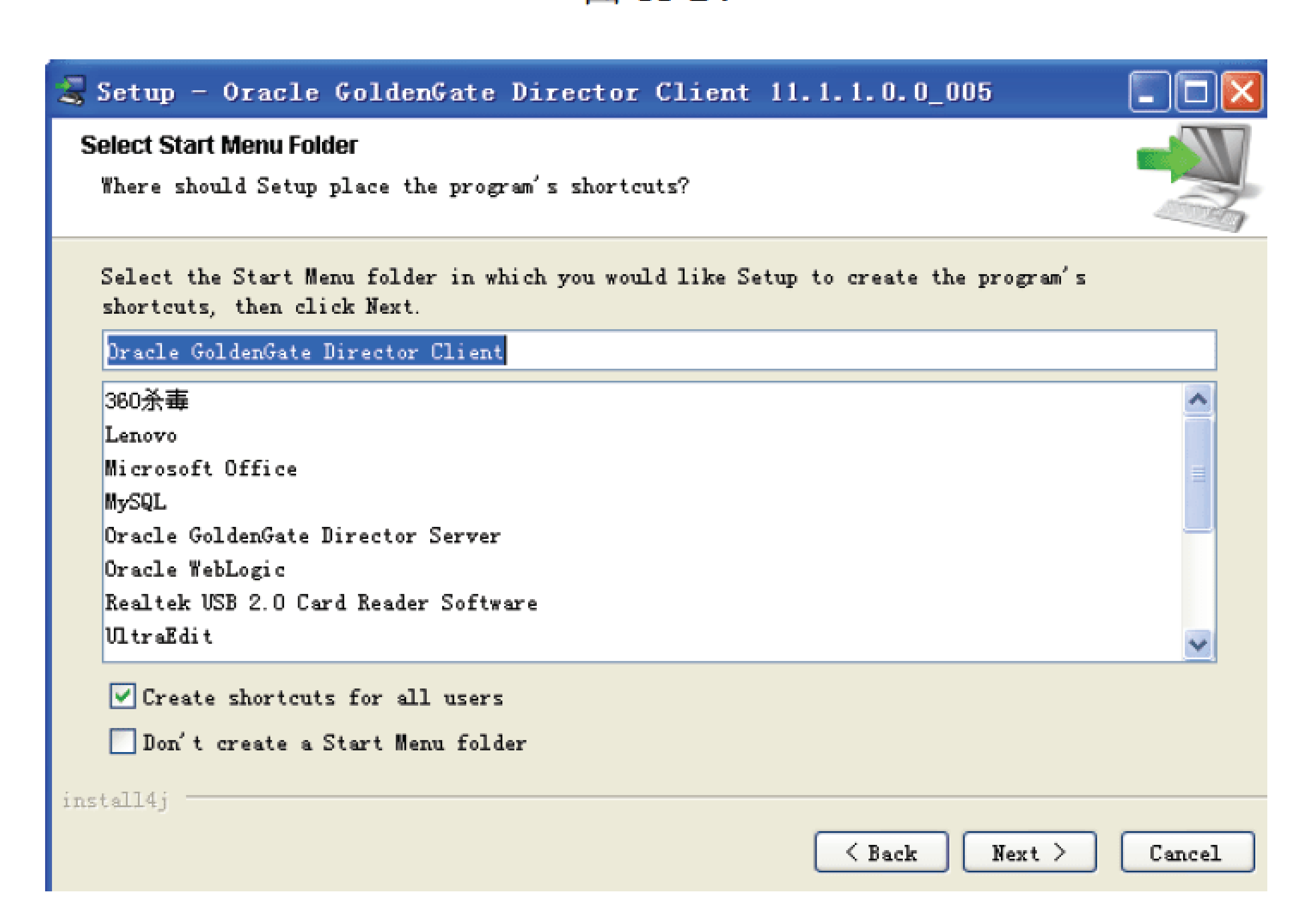


图 11-25

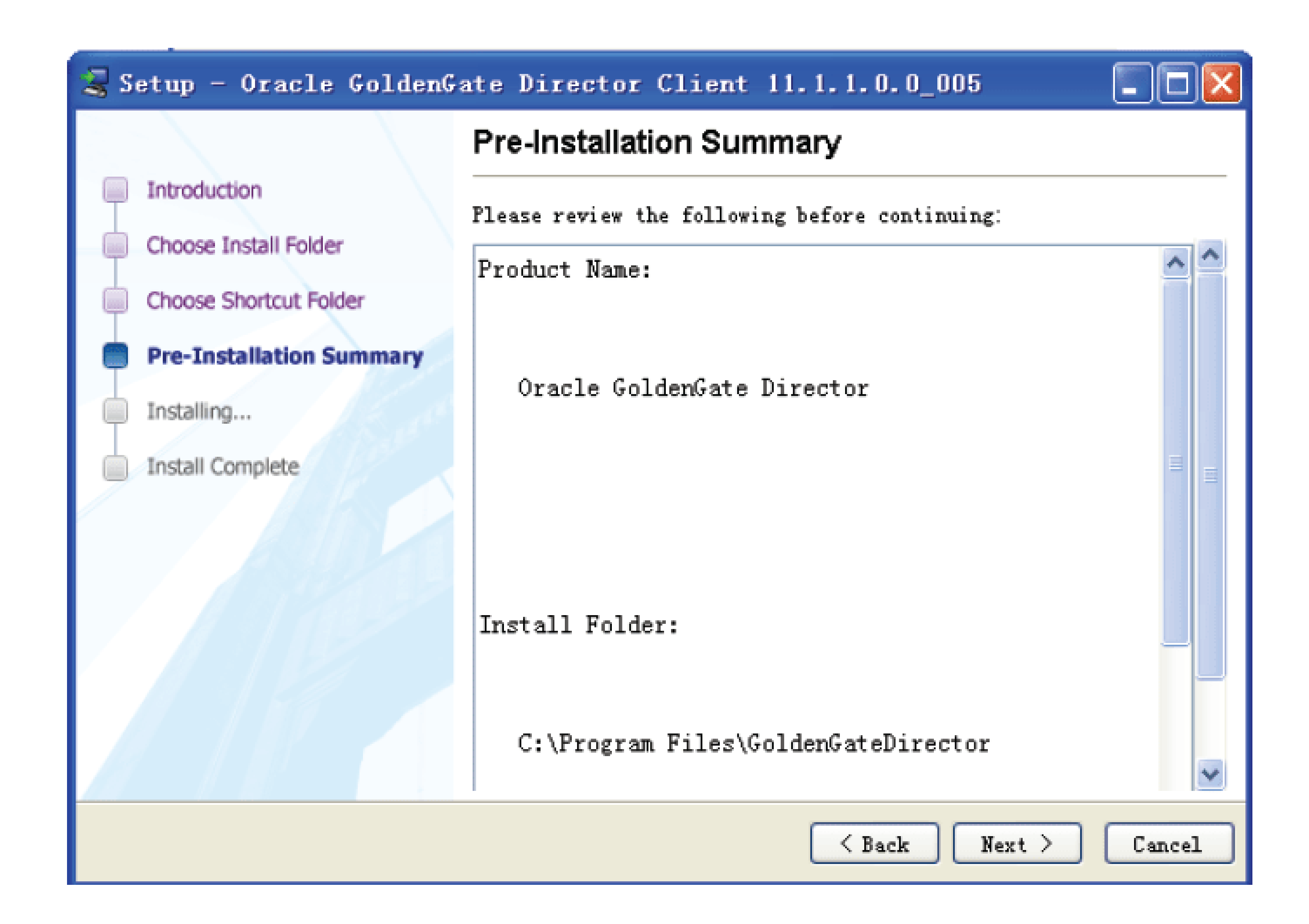


图 11-26

（6）安装进度显示，如图 11-27 所示。

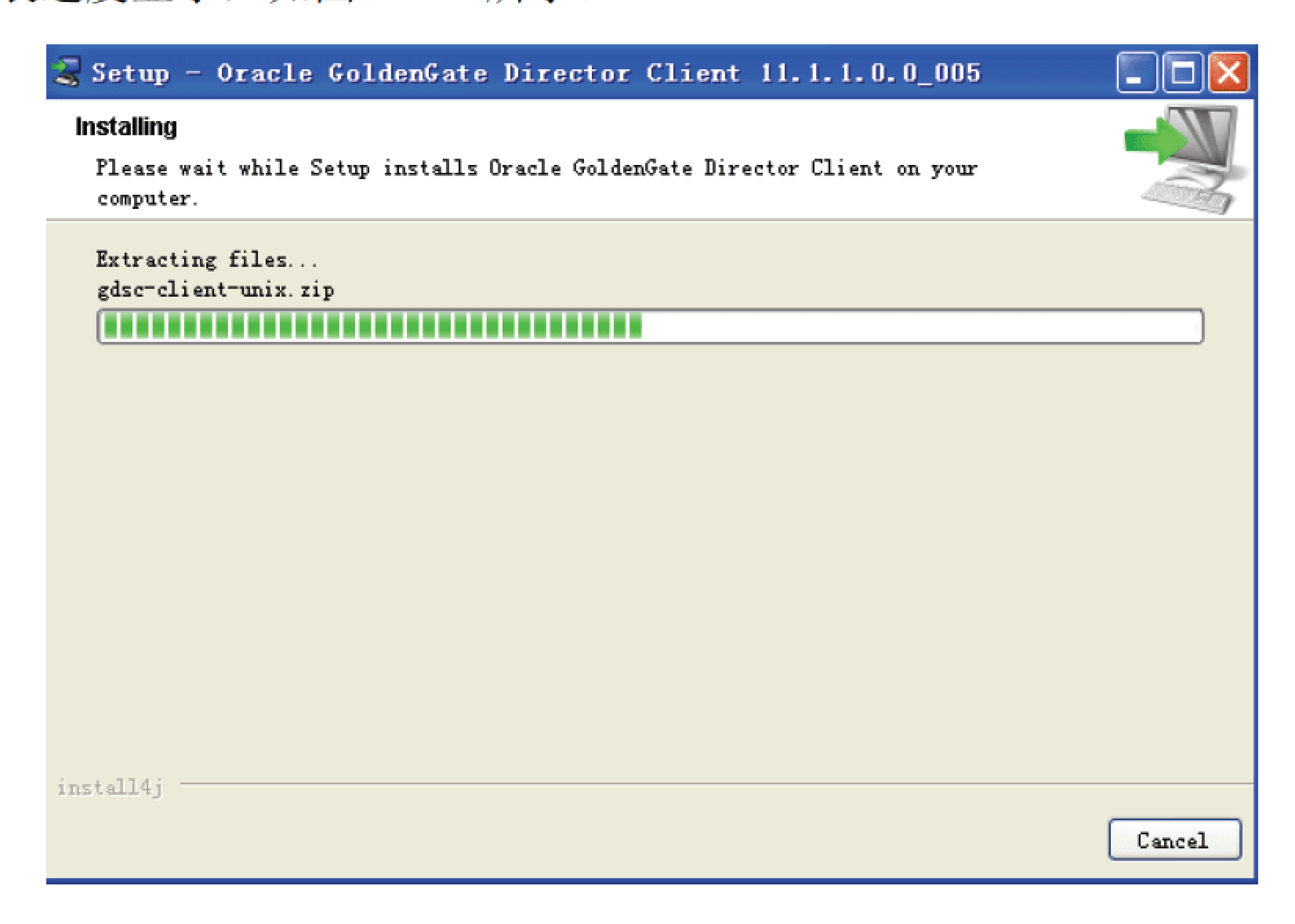


图 11-27

安装完成，如图 11-28 所示，单击 Finish 按钮完成。

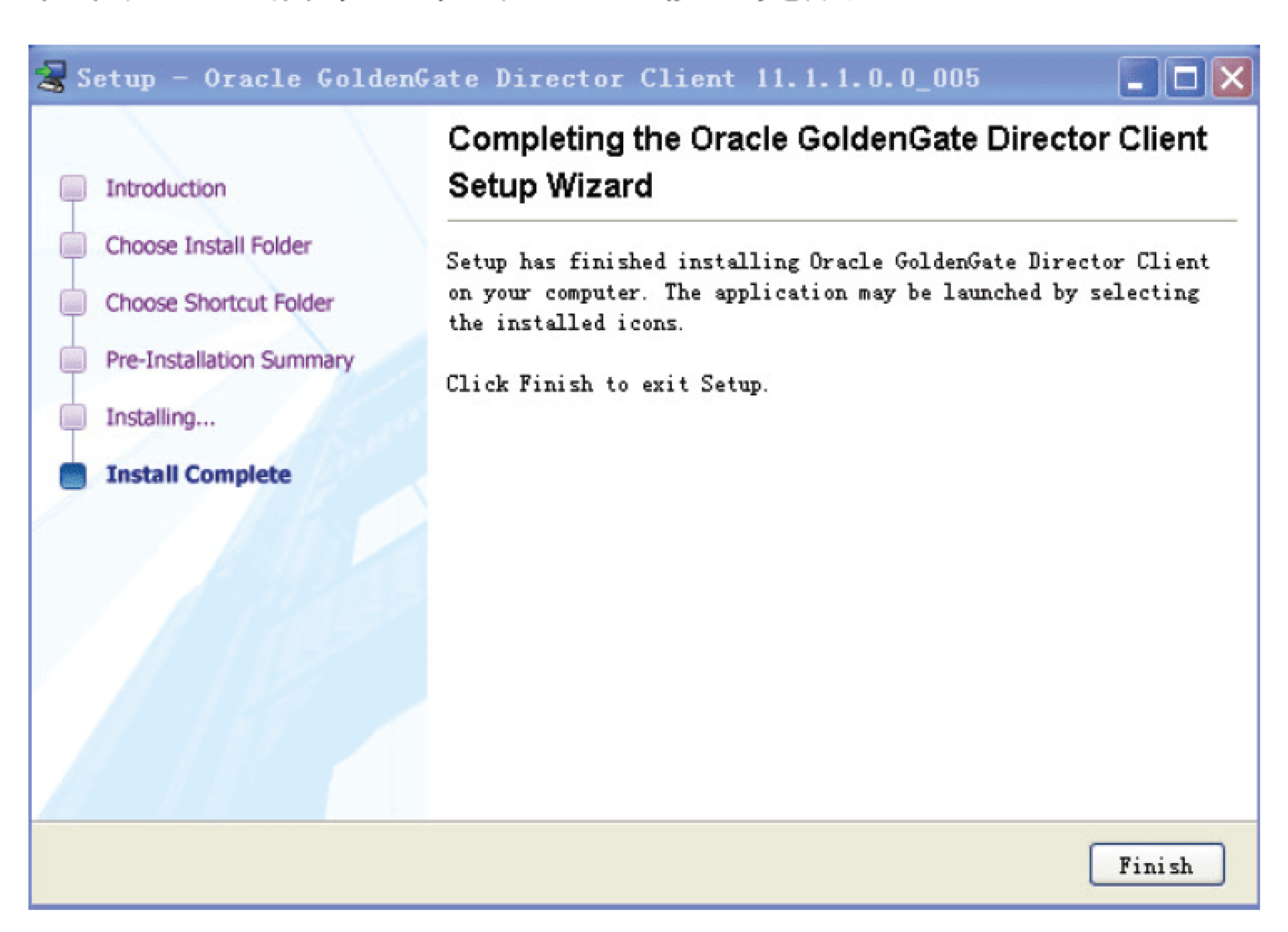


图 11-28

11.4.4　GoldenGate Director 监控配置

配置 GoldenGate Director 监控之前需要启动 Director Server，并且确保 Director database 已经启动并且正常运行。

进入安装目录，运行 startWebLogic.cmd，如图 11-29 所示。

配置被监控实例的步骤如下。

进入到 GoldenGate Director Client 安装目录，打开 GDSC Admin Tool.exe，如图 11-30

所示。首次登录的用户名和密码是 admin/admin，Server 的名字一定要加上端口号，否则它找不到端口。

```
Start Oracle GoldenGate Director
<2011-2-11 上午10时06分15秒 CST> <Notice> <WebLogicServer> <BEA-000365> <Server
state changed to RESUMING>
<2011-2-11 上午10时06分19秒 CST> <Notice> <Server> <BEA-002613> <Channel "Defaul
t" is now listening on 192.168.0.122:7001 for protocols iiop, t3, ldap, snmp, ht
tp.>
<2011-2-11 上午10时06分19秒 CST> <Notice> <Server> <BEA-002613> <Channel "Defaul
t[1]" is now listening on 192.168.50.100:7001 for protocols iiop, t3, ldap, snmp
, http.>
<2011-2-11 上午10时06分19秒 CST> <Notice> <Server> <BEA-002613> <Channel "Defaul
t[3]" is now listening on 127.0.0.1:7001 for protocols iiop, t3, ldap, snmp, htt
p.>
<2011-2-11 上午10时06分19秒 CST> <Notice> <Server> <BEA-002613> <Channel "Defaul
t[2]" is now listening on 192.168.11.1:7001 for protocols iiop, t3, ldap, snmp,
http.>
<2011-2-11 上午10时06分19秒 CST> <Notice> <WebLogicServer> <BEA-000331> <Started
 WebLogic Admin Server "192.168.50.105" for domain "domain" running in Developme
nt Mode>
<2011-2-11 上午10时06分20秒 CST> <Warning> <Server> <BEA-002611> <Hostname "comp
uter", maps to multiple IP addresses: 192.168.0.122, 192.168.50.100, 192.168.11.
1>
<2011-2-11 上午10时06分22秒 CST> <Notice> <WebLogicServer> <BEA-000365> <Server
state changed to RUNNING>
<2011-2-11 上午10时06分22秒 CST> <Notice> <WebLogicServer> <BEA-000360> <Server
started in RUNNING mode>
```

图 11-29

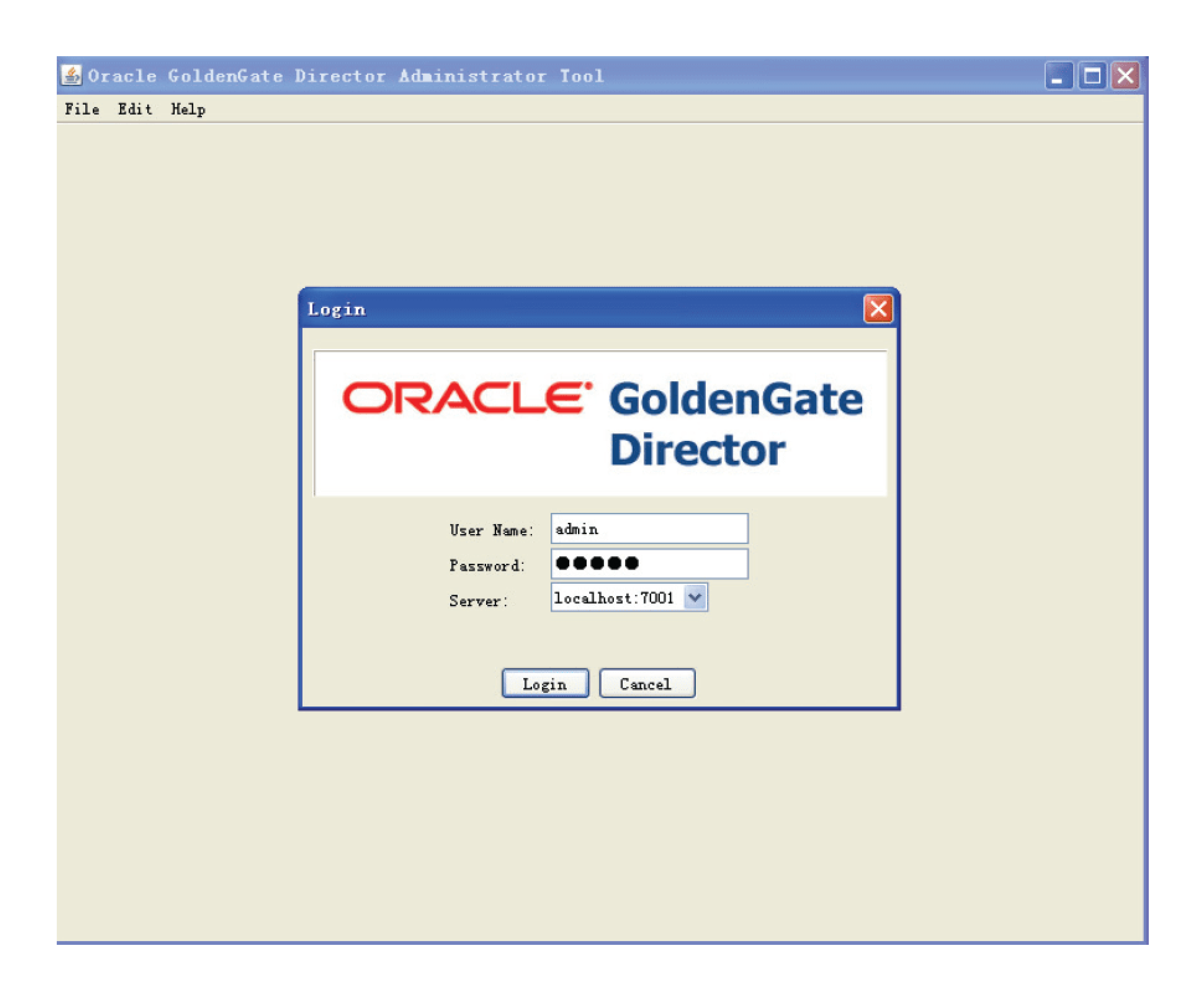


图 11-30

（1）进入到 Director Server 管理界面如图 11-31 所示，单击 Data Sources 选项卡（一个 Data Source 代表一个 GG 实例）。

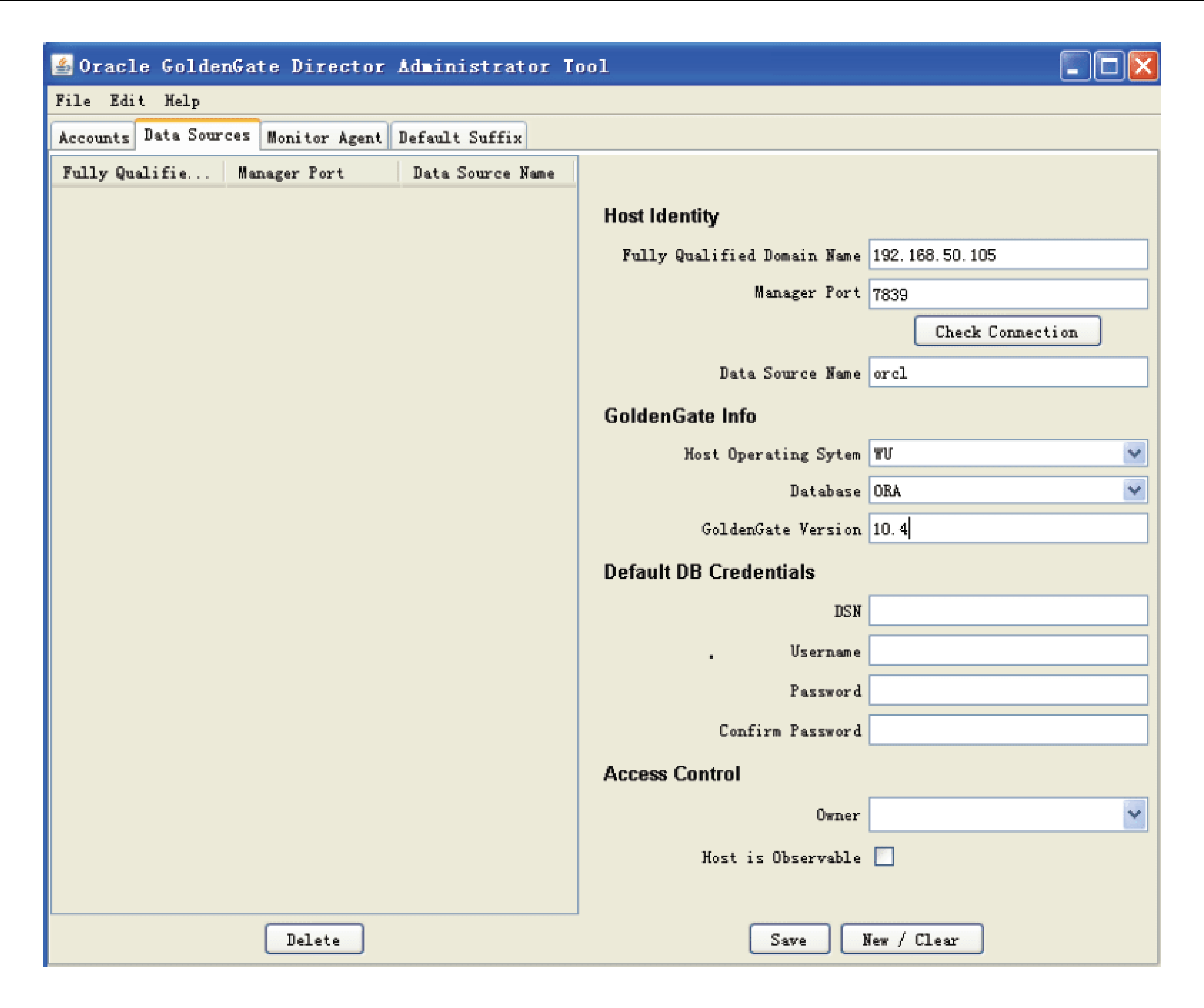


图 11-31

（2）配置需要监控和维护的 GG 实例，创建 data source 的过程如下。

① Host Identity：

- **Fully Qualified Domain Name**　GG 实例所在的服务器名。
- **Manager Port**　GG 实例 MGR 进程端口号。
- **Data Source Name**　原则上可以填写任何名称，建议制定命名规范。
- **单击 Check–Connection**　验证连接。验证成功后单击 Close 按钮，如图 11-32 所示。

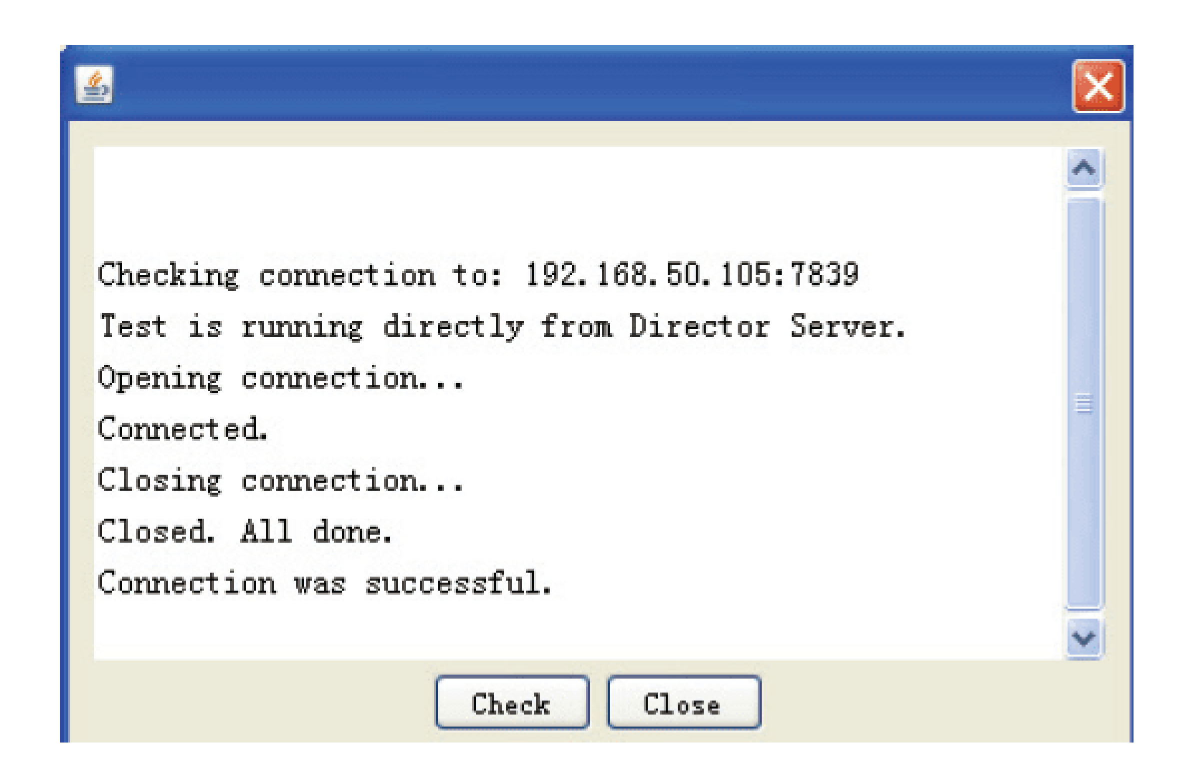


图 11-32

② GoldenGate Info：

- **Host Operating System**　选择 WU（即 Windows/Unix 意思）。
- **Database**　选择数据库类型，Oracle 数据库应选择 ORA。
- **GoldenGate Version**　10.4。
- 单击 Save 按钮，然后单击 Yes 按钮。

③ 从左侧的面板上选择刚才创建的 Data Source，填写完其他信息。

④ Default DB Credential：

- **DSN**　数据库实例名或服务名。
- **Username**　用户名。
- **Password & Confirm Password**　密码。

⑤ Access Control：

- **Owner**　选择 admin。
- 勾选 Host is Observable 选项。
- **GoldenGate Version**　10.4（需要再填写一次）。
- 单击 Save 按钮。

⑥ 关于 Data Source 的命名规范，建议采用如下格式。

<源或目标标志符>_<主机名>_<数据库类型>，比如：

示例 11-5：

```
source_node1_oracle
```

⑦ 配置完成后如图 11-33 所示。

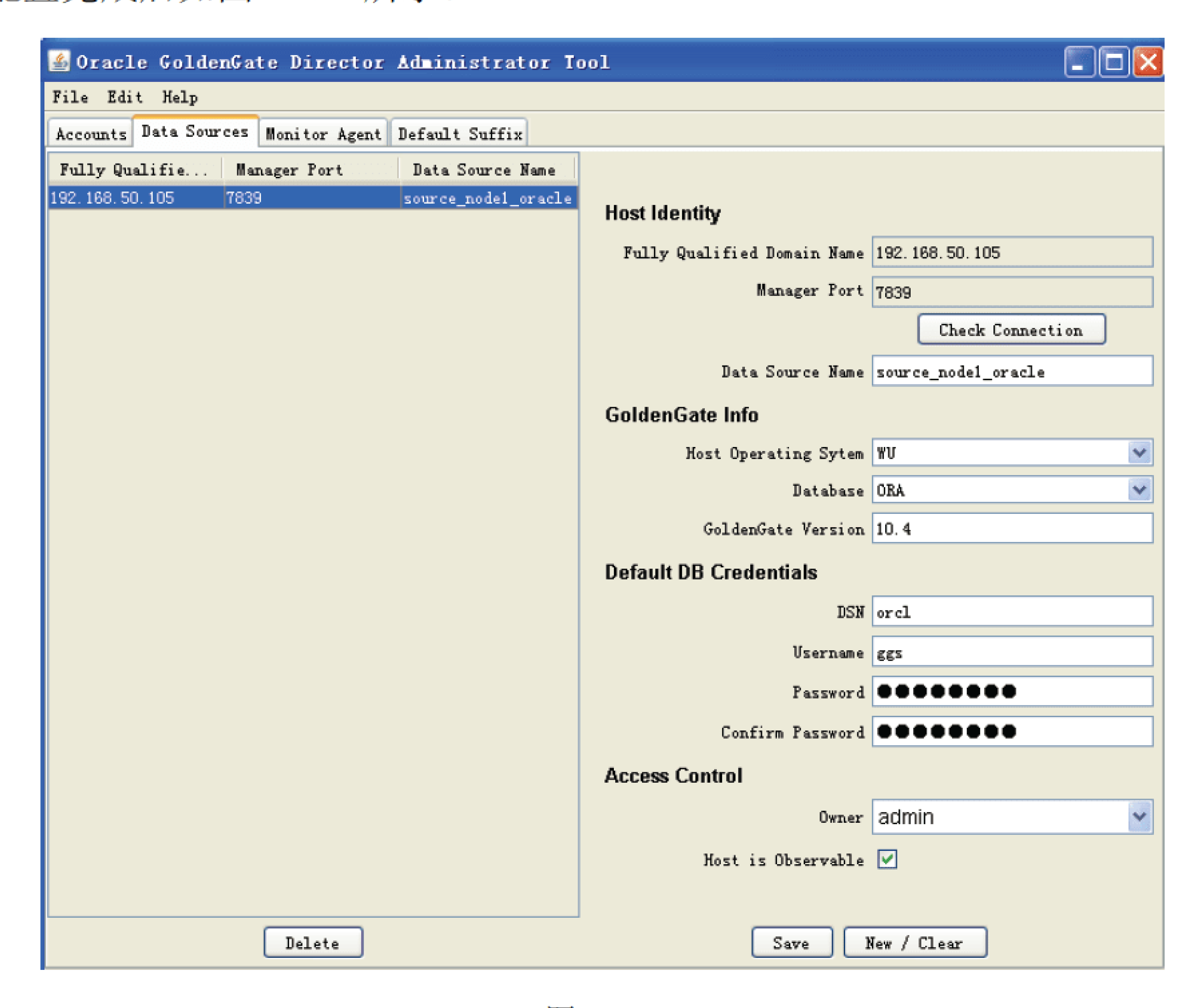


图 11-33

11.5　Web 监控界面

要进入 Director Web 界面，在浏览器输入网址 http://<systemname>:<port>/acon，比如：

示例 11-6：

```
http://localhost:7001/acon
```

<system name>是 Director Server 所在服务器的主机名或 IP 地址，<port>为安装时选择的端口号，要确保开启此端口号给 Director Server 专用。

首次进入 Director Web，需要输入用户名和密码（默认为 admin/admin），进入之后的主界面如图 11-34 所示。

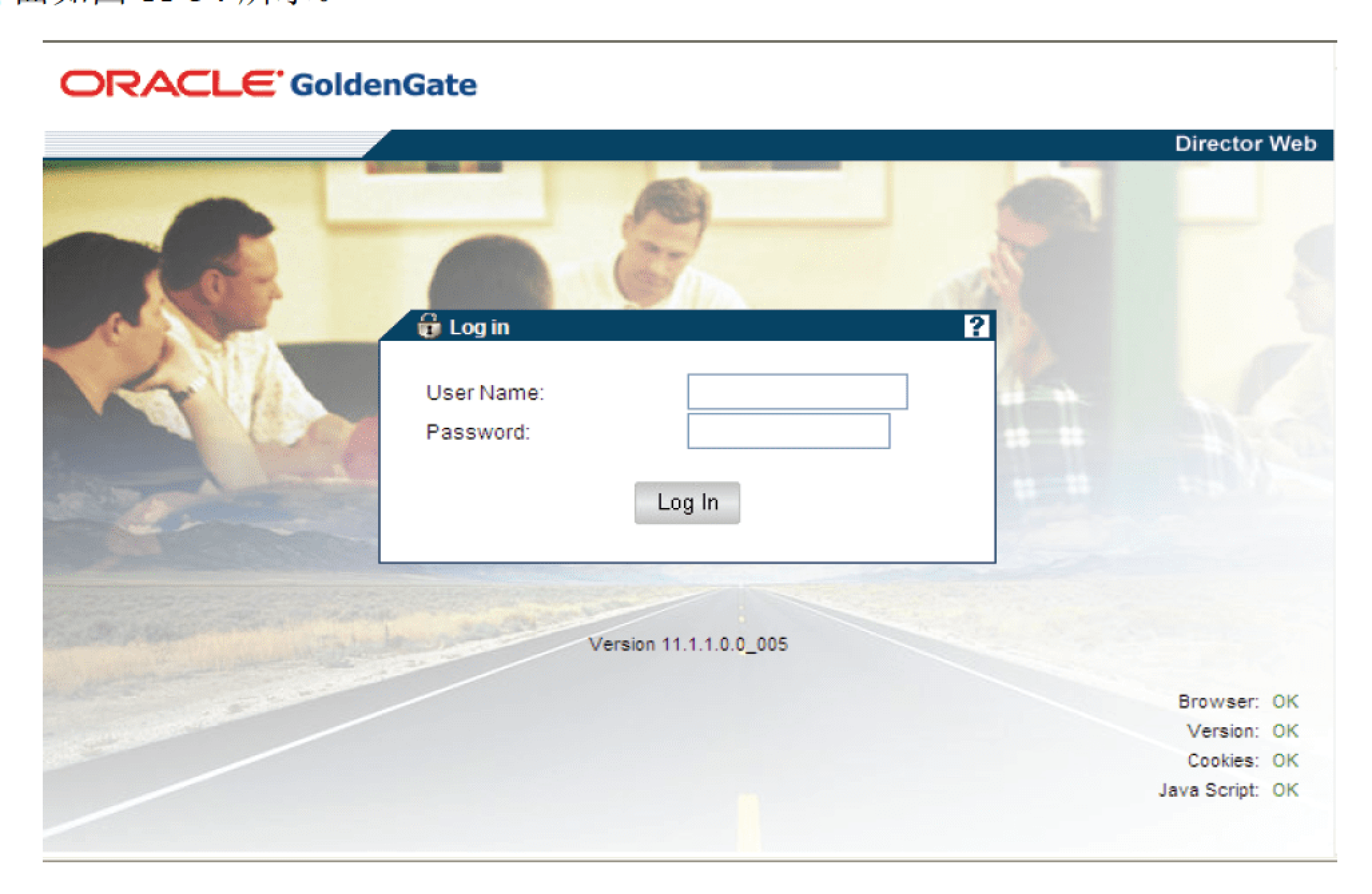


图 11-34

输入用户名和密码 admin/admin，登录到监控界面如图 11-35 所示。



图 11-35

整个主界面分为 3 个区域。

- 左侧面板包含了所有 GG 实例的列表以及一些功能链接。
- 右侧上方按照 GG 实例分组列出了每个实例上所有进程的概况。
- 右侧下方列出了所选实例的事件日志。

表示实例正在运行（MGR 进程正常运行）。

表示实例没有启动（MGR 进程已停止）。

表示进程正常运行。

表示进程正常停止。

表示进程非正常停止，需要进一步调查原因，同时会用红色的 Abended 字样标示。

11.5.1 监控进程的状态

右侧上方的面板显示了每一个进程的概要信息，点击 More Info 可以查看详细的进程信息如图 11-36 所示。

这里可以查看到进程报告文件、详细信息、历史延时情况以及被丢弃的记录。Discard File 是进程用来存放丢弃记录的文件。如果该文件中有记录，需要详细查看记录情况，分析记录被丢弃的原因，并解决相关问题。

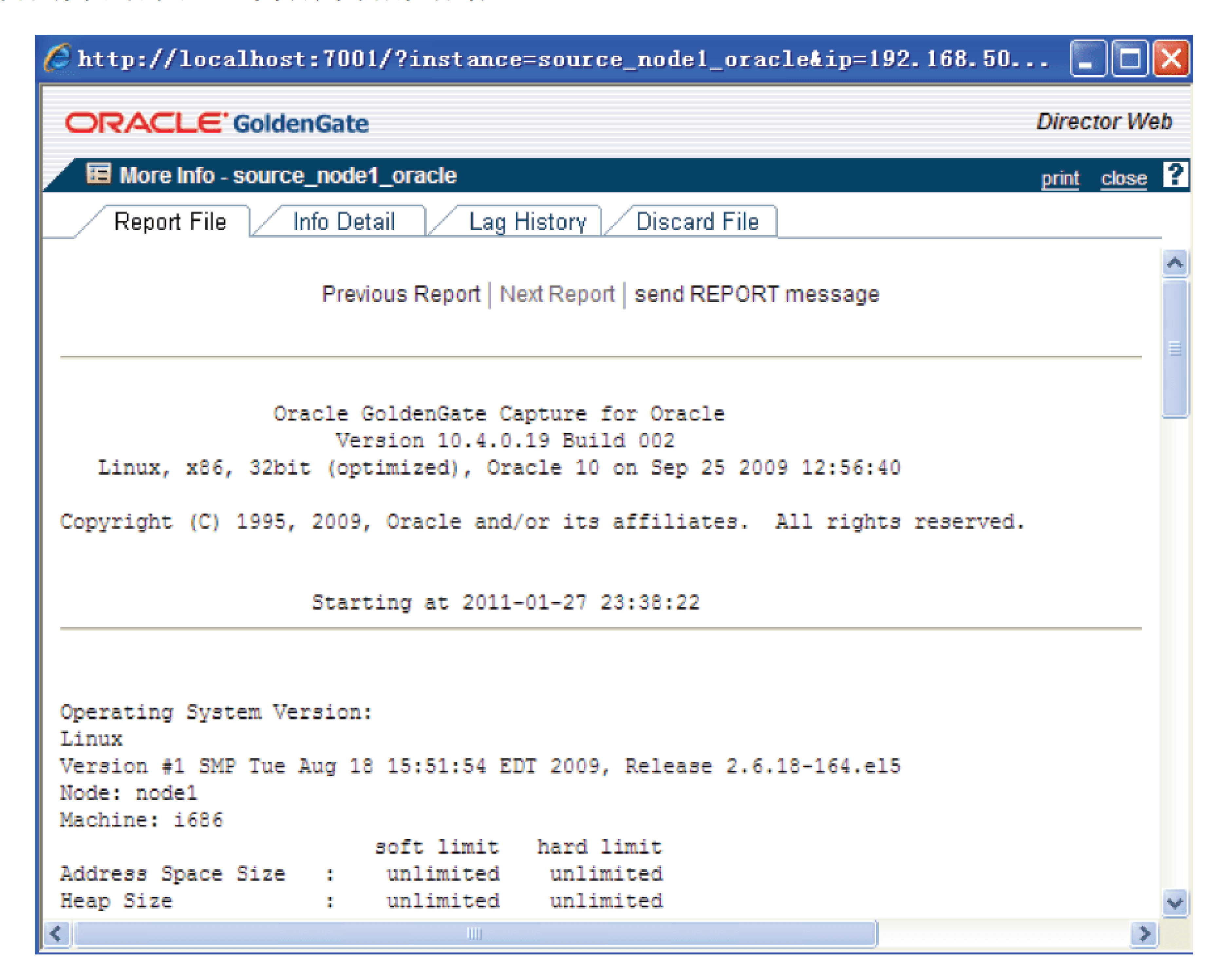


图 11-36

11.5.2 手工配置重点监控列表

默认情况下，所有进程都是按照实例分组排列的。实际监控过程中，需要把某些关联的进程组合在一起(有可能不在同一个实例)，可以通过配置监控列表的方式实现这个目标。

单击左侧面板的 Watch Lists 旁边的黑三角，然后单击 Configure Watch Lists。在新弹出的窗口中添加 Watch List。

如图 11-37 所示：

Watch List Name：为监控列表起一个名字。

Include in list：选择需要监控的进程。

Event Log Filter：选择显示哪些类型的事件消息（建议只选择 Warning 和 Error 信息）。

填写好以上信息后单击 Save 按钮。回到原来的页面，单击新添加的监控列表，可以看到右边面板显示的信息也发生了相应的变化：只有监控列表中相关进程的信息被显示了出来。

建议把一些重要且相关的进程组合到同一个 Watch List 中，以方便监控。一般来讲，同步一张表到目标需要 3 个进程：Capture 进程、Pump 进程和 Delivery 进程。其中，Capture 进程和 Pump 进程一般在同一个 GG 实例中，Delivery 进程在另一个实例中。

将同步相同表的 3 个进程组合到一起，能够较好地监控 GoldenGate 复制过程。

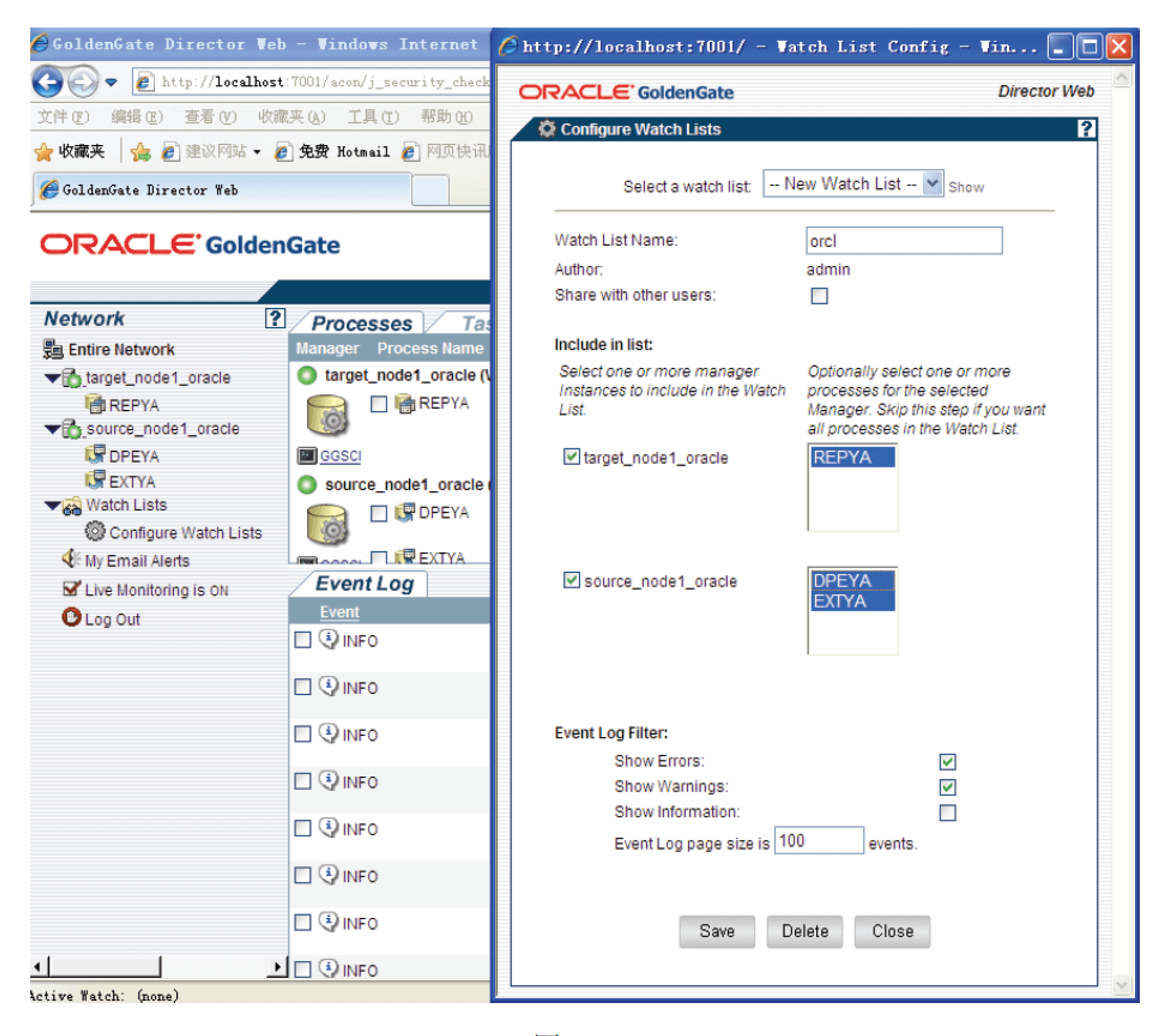


图 11-37

11.5.3　查看事件日志

主界面右侧下方是事件日志。

Info 事件只是一般的事件，如收到 stats 命令等，不会对 GoldenGate 的正常运行造成

影响。

Warning 事件是有可能对 GoldenGate 造成影响的事件，如停止 MGR 进程等。

Error 事件需要引起关注，表示有重大错误发生，已经影响到了 GoldenGate 的正常运行。

单击右上角的 Filter 按钮，把 Show Information 和 Show Un-acknowledged Only 两项勾选掉。这样，仅 Warning 和 Error 信息被显示出来如图 11-38 所示，便于发现问题。

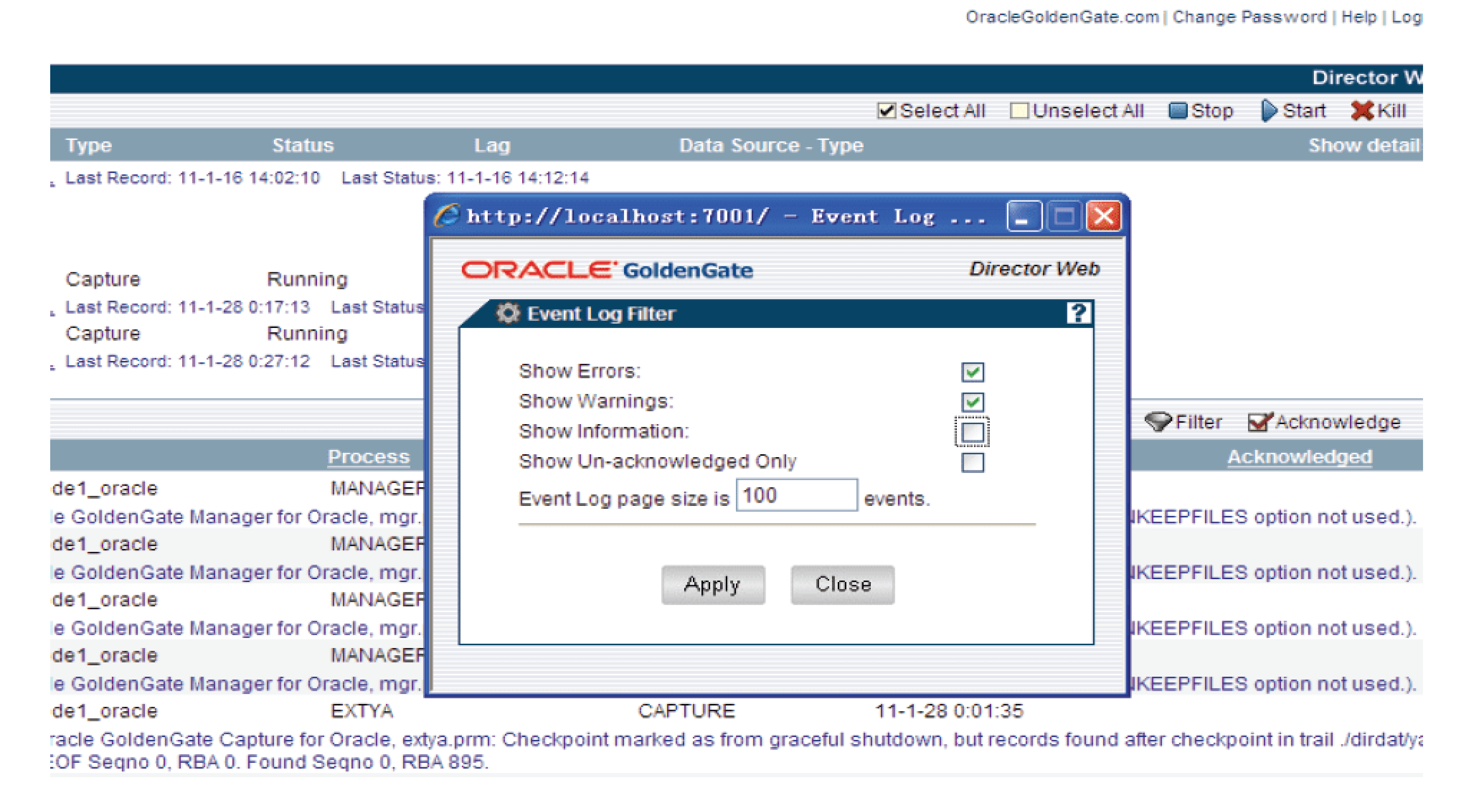


图 11-38

11.5.4　Email 告警

开启 Email 告警功能需要事先部署一个 Mail 服务器。单击主界面左侧面板上 My Email Alerts 链接，在新窗口中填写以下信息如图 11-39 所示。

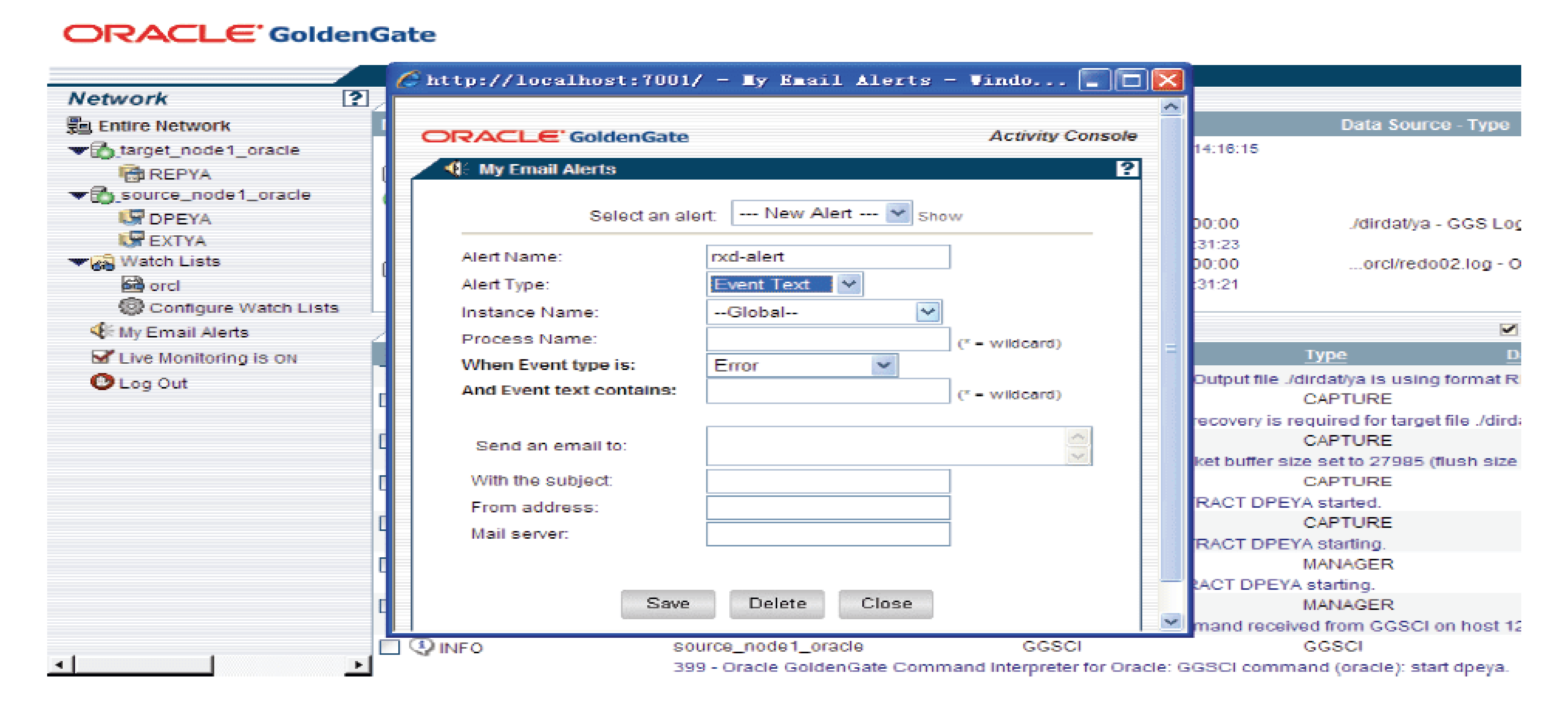


图 11-39

Alert Name：告警的名字。

Alert Type：有两种类型可供选择。

如果选择 Process Lag，则需要指定一个时限，超过这个时限会引发告警。

如果选择 Event Text，则需要指定 Event Type（Error，Warning 等），以及 Event text 中包含的文本。当 Event 消息中包含此文本时，会引发告警，支持通配符*。

Process Name：进程名称，支持通配符*。

Sent an email to：目标邮件地址。

With the subject：邮件标题。

From Address：发送方地址。

Mail Server：邮件服务器，需要事先部署。

11.5.5　运行 GGSCI 命令

选择主界面右上方某个实例，单击实例下方的 GGSCI 图标，进入命令行交互页面如图 11-40 所示。



图 11-40

在这里，可以输入各种命令，并显示输出结果如图 11-41 所示。

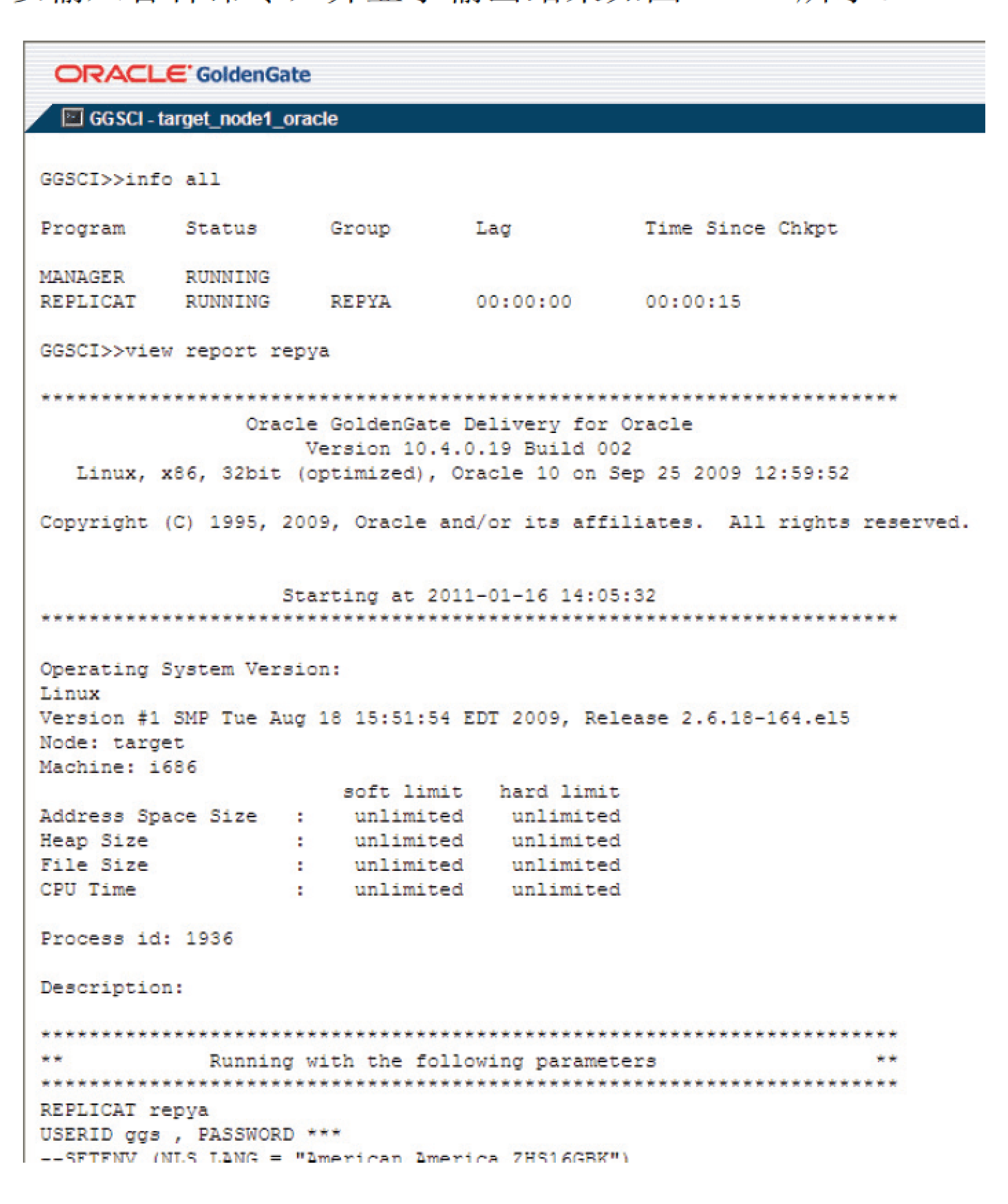


图 11-41

第 12 章　使用 GoldenGate Veridata 进行数据校验

Oracle GoldenGate Veridata 作为一个独立的产品，是一种高性能的数据对比解决方案，对在两个数据库之间进行数据复制时可能存在的差异进行确认和报告。

GoldenGate Veridata 在两端数据库保持在线的情况下进行数据对比。

12.1　GoldenGate Veridata 概述

GoldenGate Veridata 能在 Enscribe、Oracle、NonStop SQL/MP、SQL Server、Teradata 这几个平台的数据库上比较数据，比较数据时能自动匹配列数据类型。

GoldenGate Veridata 的组件包括 GoldenGate Veridata Server、GoldenGate Veridata Web、GoldenGate Veridata Repository、GoldenGate Veridata Agent、GoldenGate Veridata CLI (Command Line Interface)。

总的逻辑结构如图 12-1 所示。

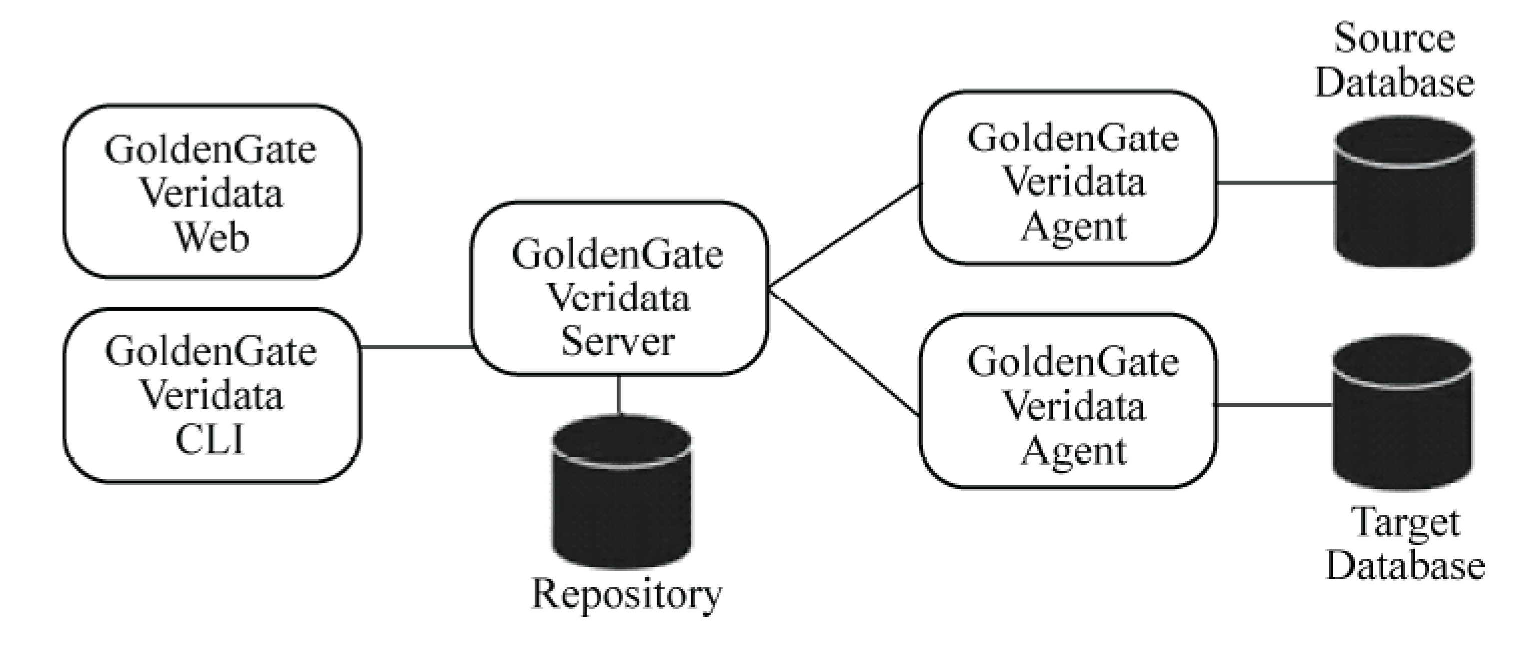


图 12-1

接下来详细介绍 GoldenGate Veridata 的各组件。

12.2　安装 GoldenGate Veridata

安装 GoldenGate Veridata 不需要 GoldenGate 复制软件的支持，如果您已经安装了 GoldenGate 复制软件的话，则需要将 GoldenGate Veridata 安装到不同的路径。

12.2.1　安装 GoldenGate Veridata 系统需求

1. GoldenGate Veridata Agent 对系统的需求

一般需要在每个要比较的数据库上安装 Agent，其中 Agent 又分为 Java-Agent 与 C-Agent。

对于 Java-Agent，需要系统安装 JRE 或者 JDK，最低版本为 1.5。

Java-Agent 对数据库的需求：

- **Oracle**　确定监听已正确配置并运行。
- **SQL Server**　一个固态的 TCP/IP 端口已配置并且没被占用。
- **Teradata**　需要知道 hostname 与数据库使用的端口，安装 Java-Agent 之前需要上官网下载相应的 JDBC 驱动。

C-Agent 对系统的需求：

C-Agent 对 Oracle 数据库是可选的，但是在同构数据库的比较上它可以提供更好的性能，对于 Oracle 数据库，C-Agent 需要用到 Oracle 的运行支持库，所以在 Oracle 数据库平台安装 C-Agent，首先需要确认环境变量有相应的路径，比如 libpath。

Agent 需要至少 1GB 的内存空间，至少 200M 的磁盘空间，主要用来进行数据的比较，同时需要有相应的数据库权限。

2. GoldenGate Veridata Server 对系统的需求

需要安装 Microsoft Visual C++ 2005 Redistributable 包。

需要一个数据库来存储 GoldenGate Veridata Web 产生的信息及环境属性。支持的数据有 MySQL、Oracle、SQL Server。

另外，至少 30M 的内存空间，一定的磁盘空间，还有相应的数据库权限。

12.2.2　安装 GoldenGate Veridata 代理

在对数据进行比较时，首先需在要比较的数据库上安装 Agent 软件。

1. 安装 GoldenGate Veridata Java 代理

在 UNIX 系统上安装 Java Agent 相对较简单。

（1）创建安装目录。

（2）下载安装介质。

（3）unzip 解压包。

（4）tar 解压 tar 包。

（5）配置属性文件，一般选默认就可以，其中需要配置的 you server.port, database.url, server.driversLocation, server.jdbcDriver, database.transaction.isolation。

Windows 下 Agent 的安装在图形向导的帮助下，单击下一步很快就可以完成安装。

2. 安装 GoldenGate Veridata C 代理

在 UNIX 系统安装 C 代理的步骤大致如下。

（1）创建安装目录。

（2）下载安装介质。

（3）将介质上传到服务器。

（4）解压 zip 包。

（5）解压 tar 包。

（6）从 Agent 的子目录，运行 GGSCI。

（7）在 GGSCI 命令行下，用 create subdirs 创建工作目录。

（8）编辑管理进程参数 edit params mgr。

（9）编辑管理进程用的端口。

（10）编辑动态端口范围（可选）。

（11）保存参数文件。

（12）启动管理进程，用 start manager 命令。

（13）用 info mgr 查看管理进程状态信息。

在 Windows 系统安装 C-Agent 与在 UNIX 系统安装 C-Agent 类似。

12.2.3 安装 GoldenGate Veridata 服务端

GoldenGate Veridata Server 是 Veridata 非常重要的一个组件，它主要负责：

- 调整 Veridata 执行的任务。
- 对行进行排序（可选）。
- 比较数据。
- 确认不一致的数据。
- 产生 report 文件。

1. 使用 Oracle 数据库安装服务端及 Web 组件

安装 GoldenGate Veridata Server 之前，确认作为 Repository 的 Oracle 数据库已正确配置好。下面以 Windows 下的 Oracle 数据库为例演示安装过程。

（1）启动作为 Repository 的 Oracle 数据库。

（2）上官网下载安装介质。

（3）运行.exe 文件如图 12-2 所示。

（4）选择安装路径如图 12-3 所示。

（5）按照图形向导配置端口，如图 12-4 所示。

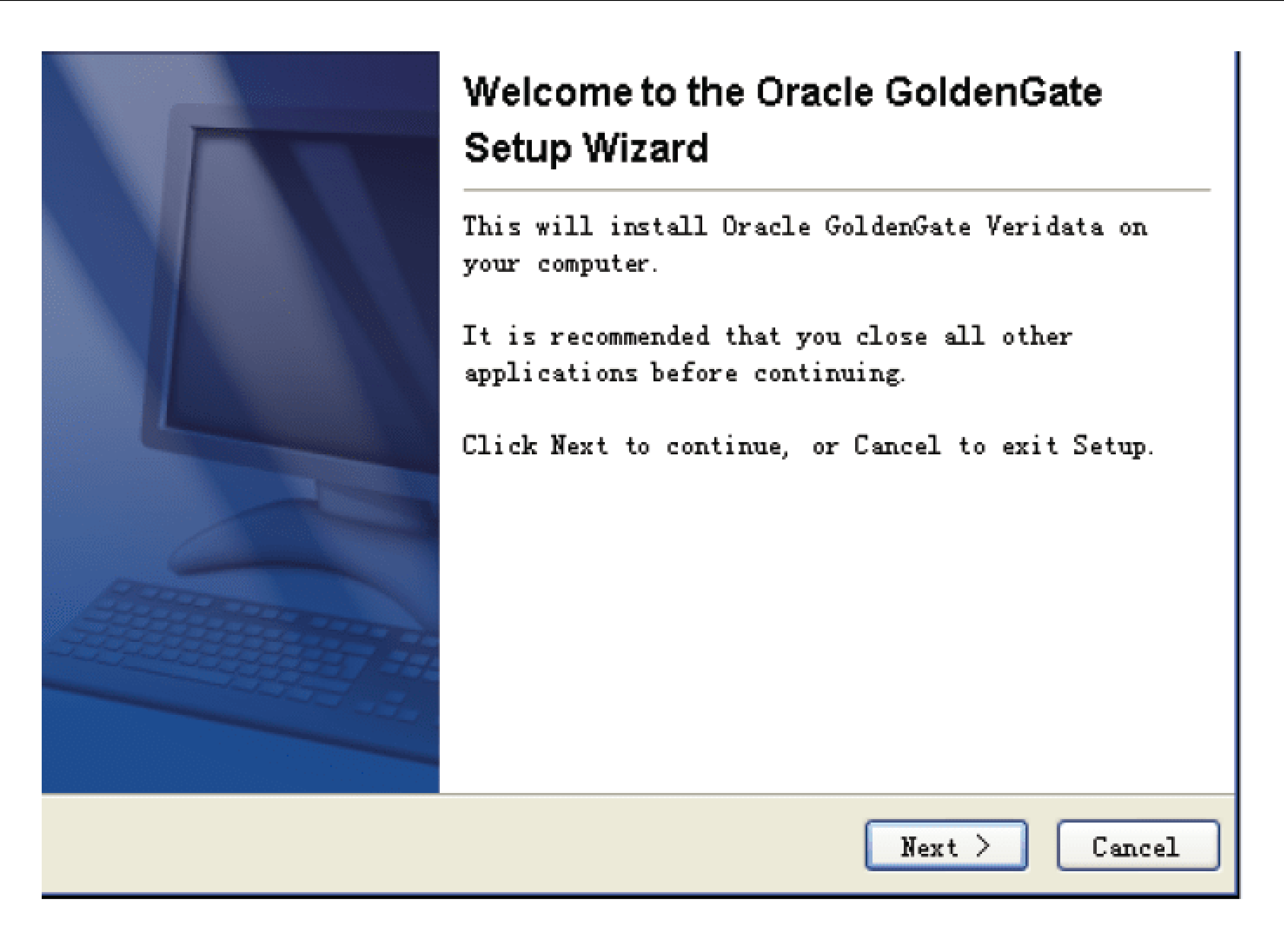


图 12-2

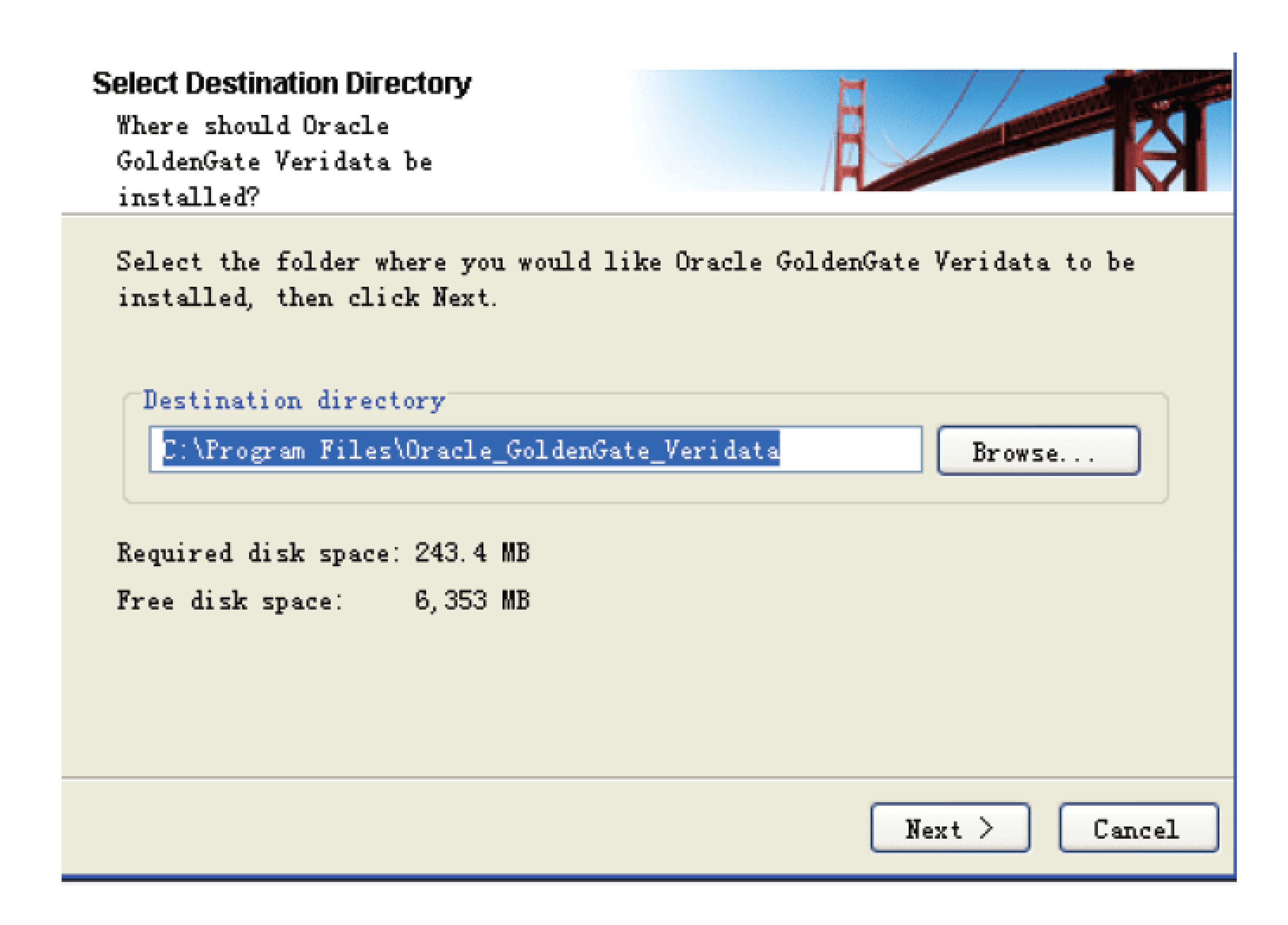


图 12-3

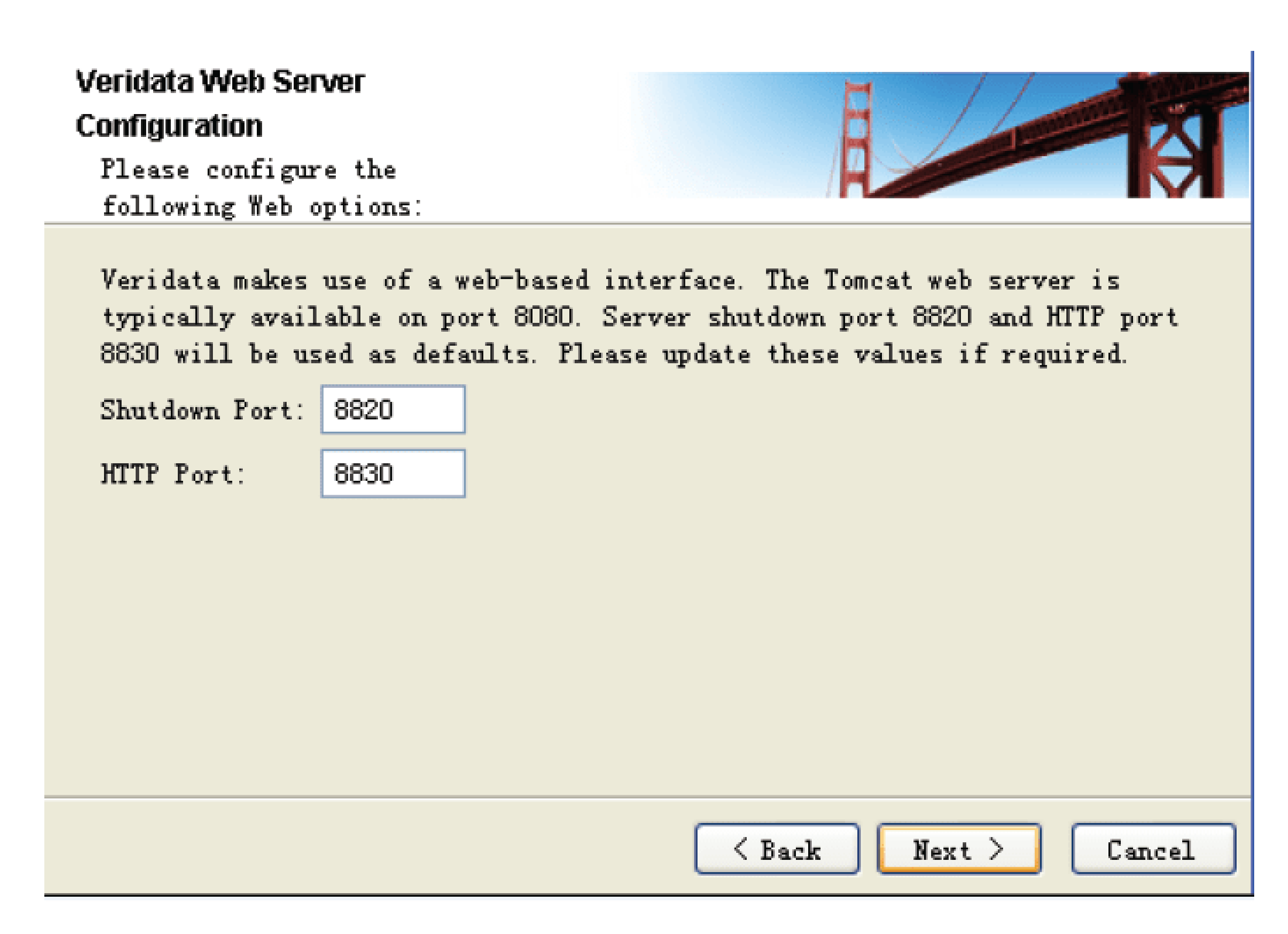


图 12-4

（6）选择 Repository 数据库的类型（这里选择 Oracle），如图 12-5 所示。

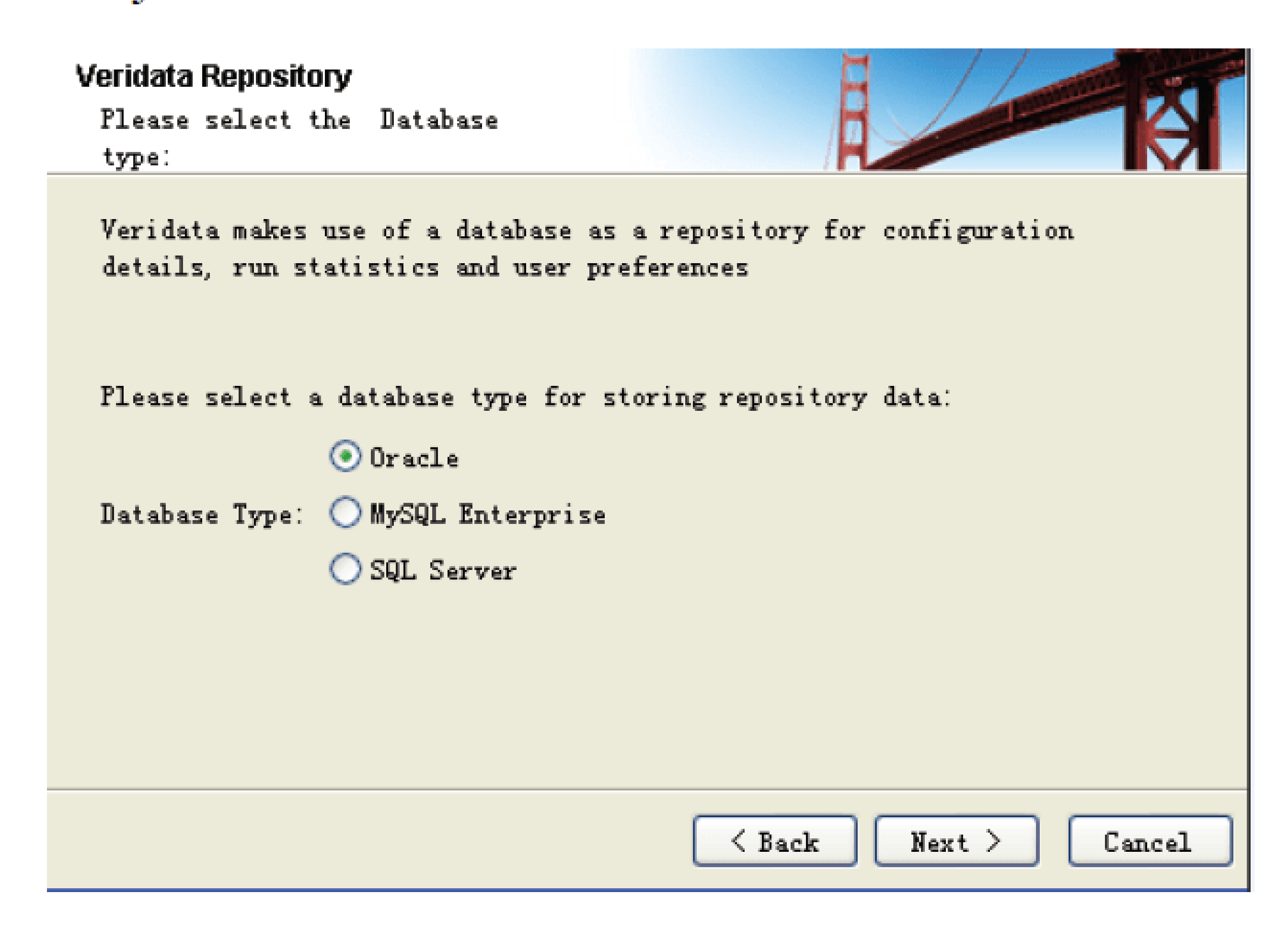


图 12-5

（7）配置 Veridata Repository，如图 12-6 和图 12-7 所示。

图 12-6

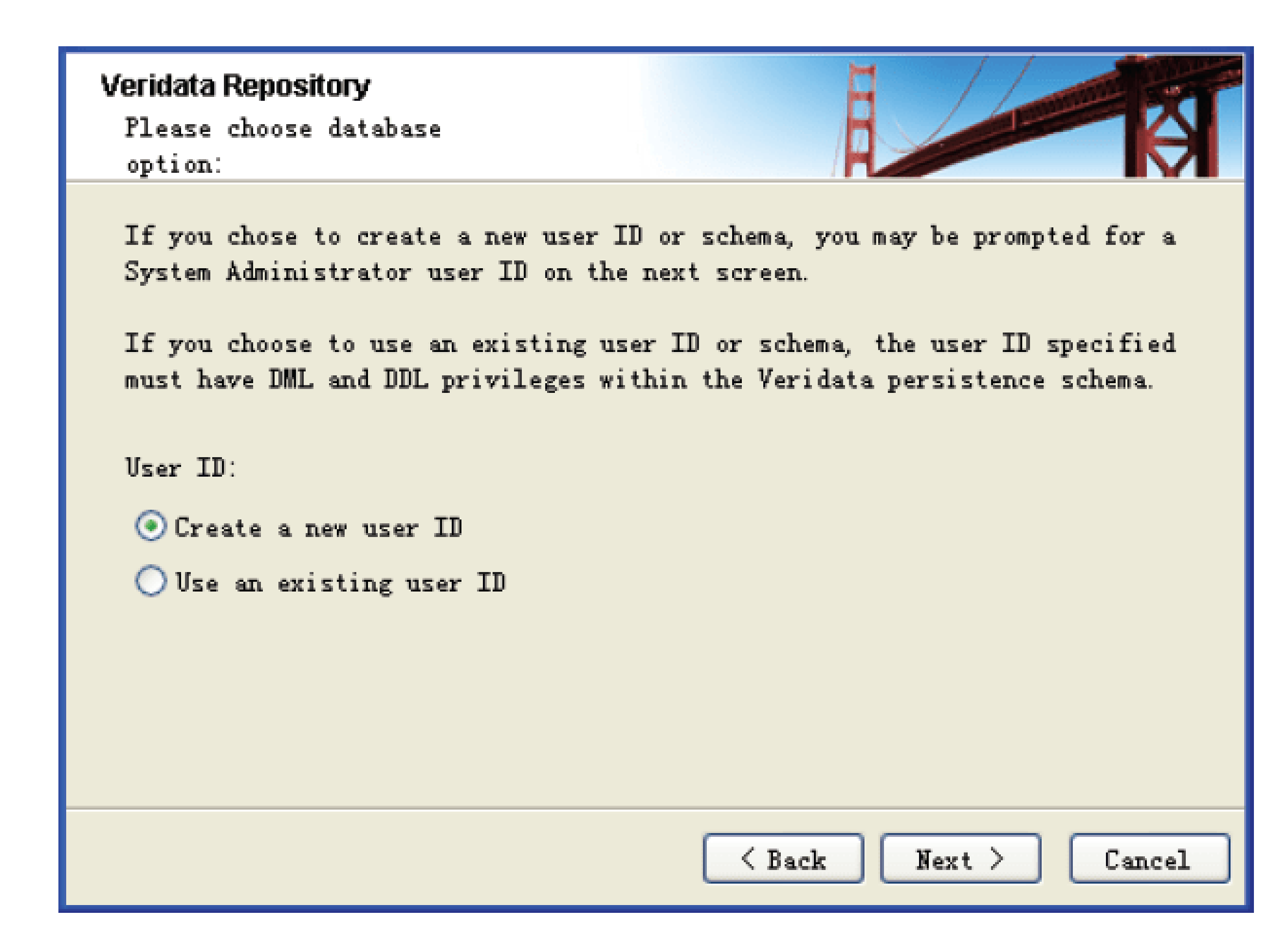


图 12-7

（8）配置 Veridata Repository 参数，如图 12-8～图 12-11 所示。

在数据库创建一个 Veridata 账号，用于记录数据比较的信息，使用的数据库表空间这里默认选择 users 表空间，临时表空间为 temp。

Veridata Repository
Please configure the following Veridata Repository details:

Please specify a User ID and Password for the Veridata Repository:

Oracle User ID: veridata
Oracle Password: ******
Confirm Password: ******

< Back　Next >　Cancel

图 12-8

Veridata Oracle Tablespace
Please specify the following Tablespace detail:

In order to create / update the Oracle Schema for the repository, please specify Tablespaces to be used:

Default Tablespace Name: users
Temporary Tablespace Name: temp

< Back　Next >　Cancel

图 12-9

Veridata Server Configuration
Please configure the following options:

Please specify the Veridata Server listening port below.
Server Communication Port: 4150

< Back　Next >　Cancel

图 12-10

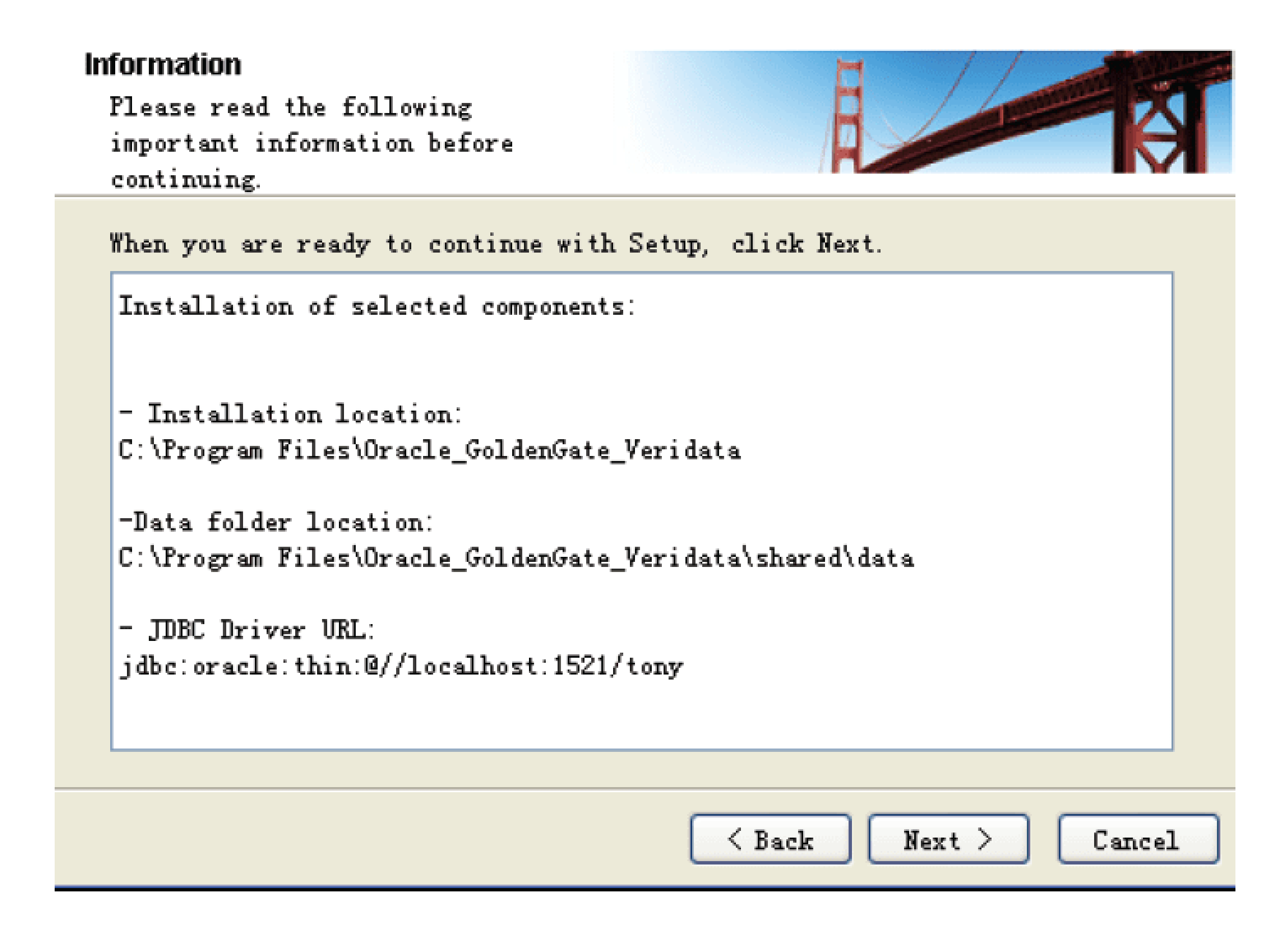


图 12-11

（9）最后单击 Next 按钮就可以到安装完成的界面。

2. 使用 MySQL 数据库安装服务端及 Web 组件（可选）

当选择 MySQL 数据库作为 Veridata Repository 时，就需要用到此向导。

安装过程与 Oracle 数据库作为 Veridata Repository 的过程较类似，首先确定 MySQL 数据库已经正常运行，接着在图形向导的帮助下，依次单击 Next 按钮即可完成安装。

3. 使用 SQL Server 数据库安装服务端及 Web 组件（可选）

当选择 SQL Server 数据库作为 Veridata Repository 时，就需要用到此向导。

安装过程与 Oracle 数据库作为 Veridata Repository 的过程较类似，首先确定 SQL Server 数据库已经正常运行，接着在图形向导的帮助下，依次单击 Next 按钮即可完成安装。

12.3 配置 GoldenGate Veridata 的安全属性

由于 Veridata 是直接比较应用里的数据，对于一些敏感数据，保障其安全性则非常有必要的。当进行数据比较时，一定要注意 Veridata 的安装目录里的一些数据文件，比较结果文件以及 Veridata Web 的界面信息，以防泄漏一些敏感信息。

一般只需要控制好比较的结果文件的权限，不让没有权限的账号查看该文件，以及保管好 Veridata Web 登入的账号密码就能很好地控制 GoldenGate Veridata 的安全属性。

12.4 运行 GoldenGate Veridata 程序进行数据比较

启停 Veridata 各组件的顺序可以是任意的。关于运行 GoldenGate Veridata，然后进行数

据的比较，得到最终需要的结果，才是最关键的。

12.4.1　启动 C 代理及 Manager

C-Agent 是跟 Veridata Server 自动启动的，但是在这之前，要确认与 Agent 关联的数据库已启动，管理进程已运行。其中运行管理进程的命令为：

进入 GGSCI 界面行：

示例 12-1：

```
GGSCI > start manager
```

或者：

示例 12-2：

```
GGSCI > stop manager
```

12.4.2　启动和停止基于 Java 组件

GoldenGate Veridata Server 与 GoldenGate Veridata Web 都是基于 Java 语言开发的组件，而 Agent 也有基于 Java 语言开发的版本，其中 Java 版本的 Agent 可以运行在除了惠普的 nonstop 的任何 Veridata 所支持的平台上。

启动和控制 Java 版本的 Agent 也较简单。

UNIX 平台，到相应的安装目录，找到 agent.sh，然后可以用：

示例 12-3：

```
shell > agent.sh {start | run| stop}
```

就可以运行或者停止 Agent。

Windows 平台，则可以在程序里找到 start agent，鼠标单击即可，或者在 cmd 命令行，用：

示例 12-4：

```
agent.bat{start | run|stop}
```

即可。

12.4.3　连接到 GoldenGate Veridata Web 界面

启动完 Veridata Agent，然后启动完 Veridata Server，就可以启动 GoldenGate Veridata Web，进入 GoldenGate Veridata Web 界面了。

在 Windows 系统，在开始菜单里直接就有启动菜单，直接单击即可，如图 12-12 所示。

或者直接在浏览器中输入 http://<hostname>:<port>/veridata

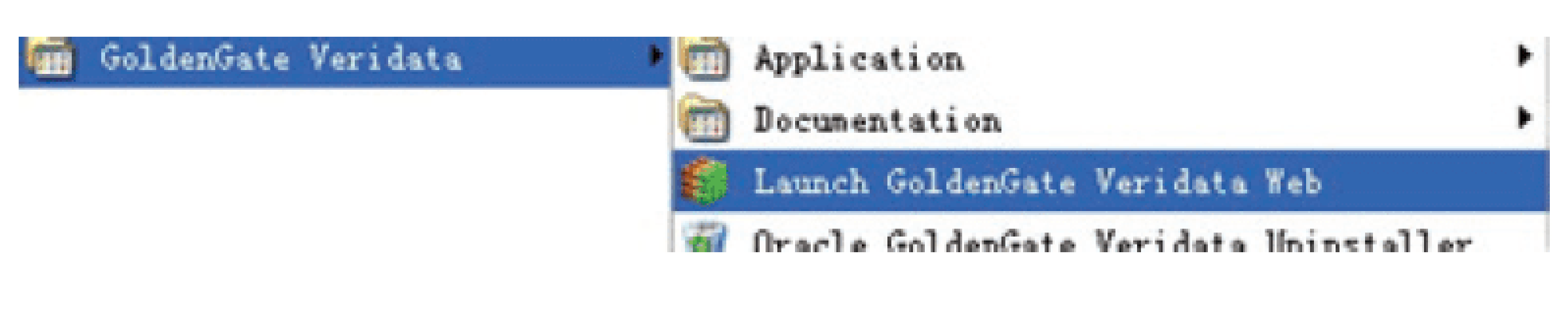


图 12-12

例如：

示例 12-5：

```
http://localhost:8830/veridata/welcome.jsf
```

都可以进入 Web 界面，如图 12-13 所示。

ORACLE GoldenGate Veridata

Please provide login credentials:

Username: wayneshen730

Password: ●●●●●●●●●

Login

图 12-13

进入 Veridata Web 界面之后，还需要配置一些比较对象（GoldenGate Comparison Objects），Veridata 才能正确地工作运行。

（1）需要配置源数据的连接，需把要比较的两个数据库的信息配置完整。

如图 12-14 所示，进入 Web 界面后，单击左下角的 Connection Configuration 如图 12-14～图 12-17 所示。

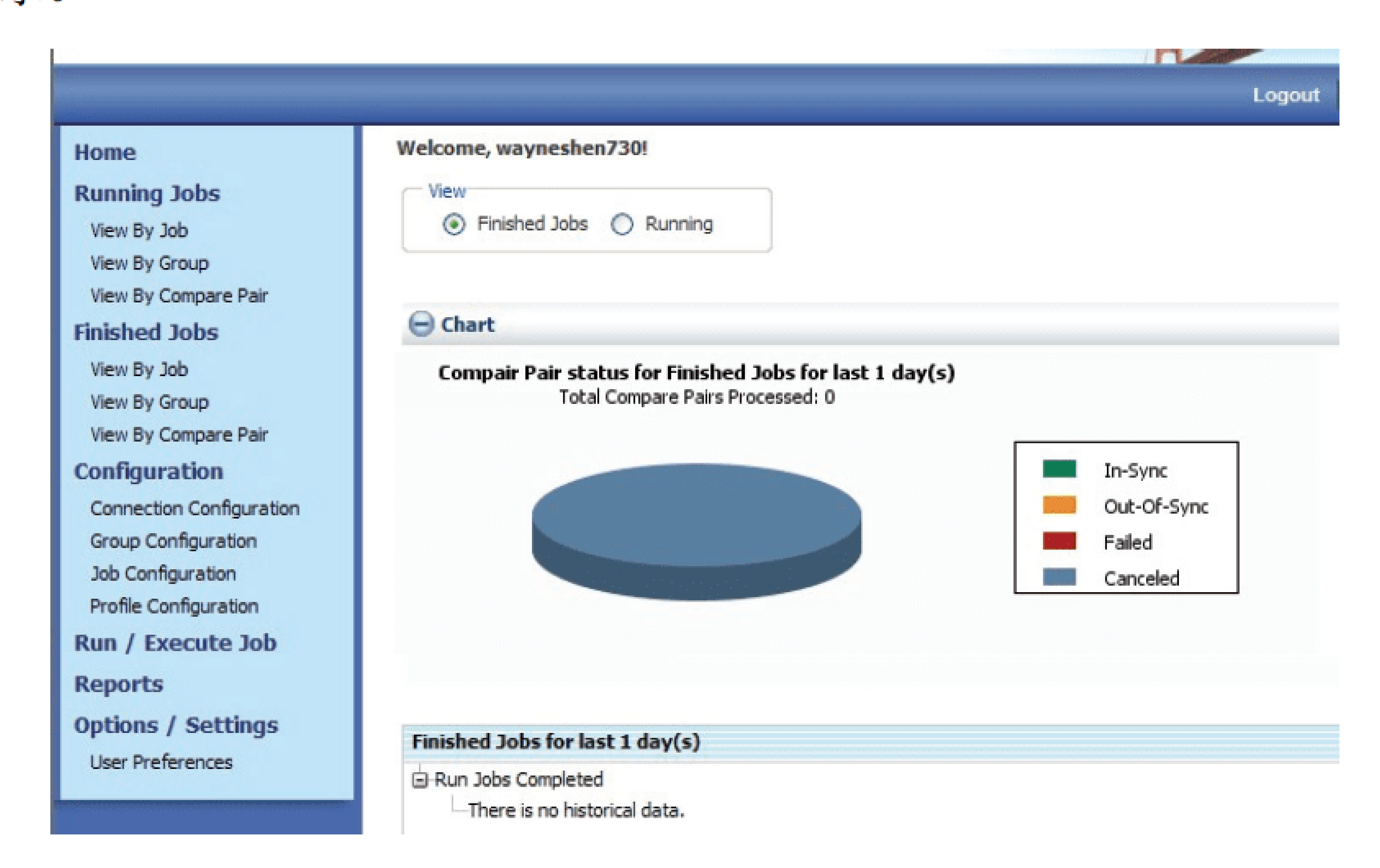


图 12-14

（2）配置要比较的组信息（一般即生产端与容灾端），然后让它们比较数据的一致性如图 12-18、图 12-19 所示。

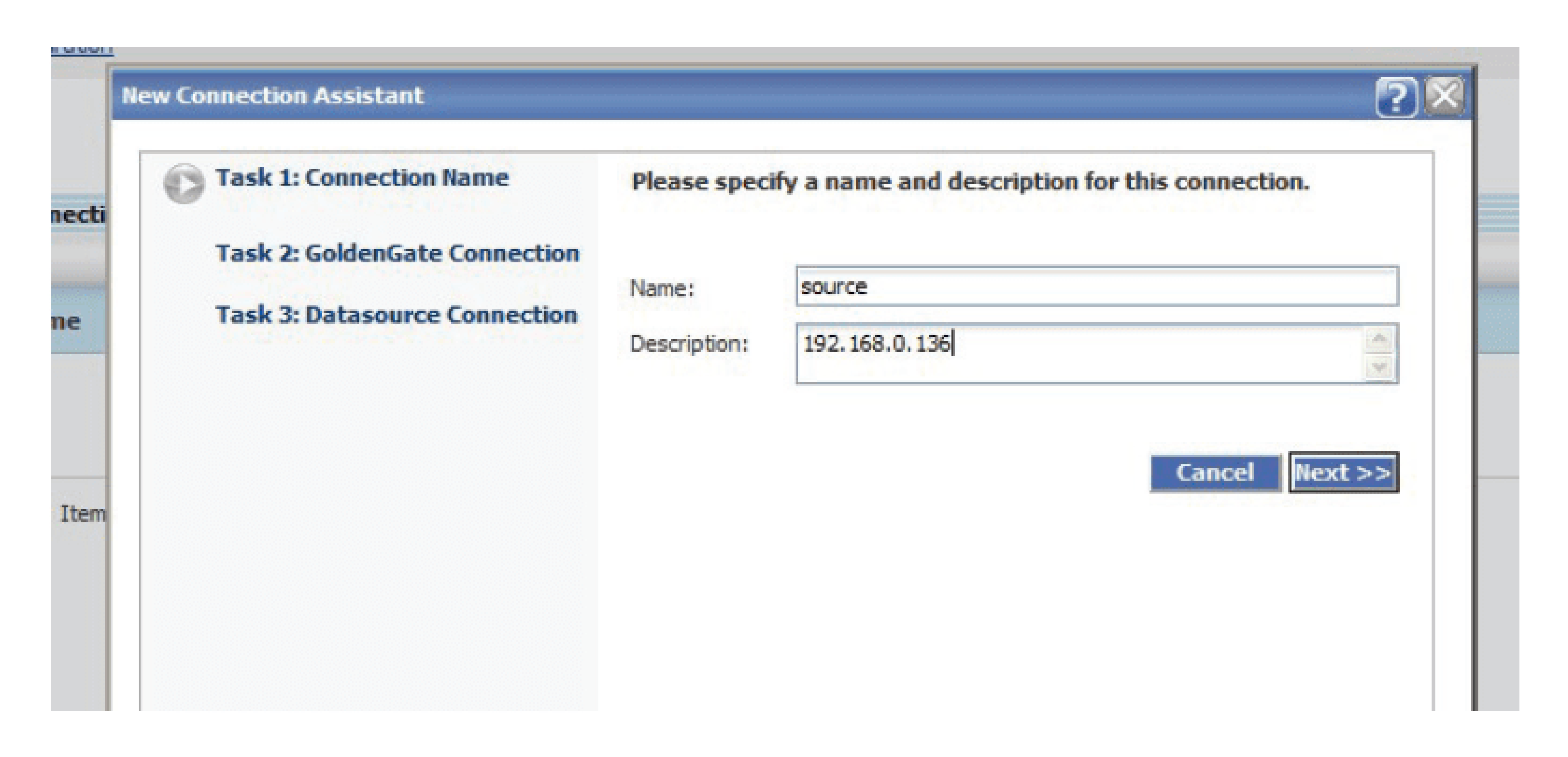


图 12-15

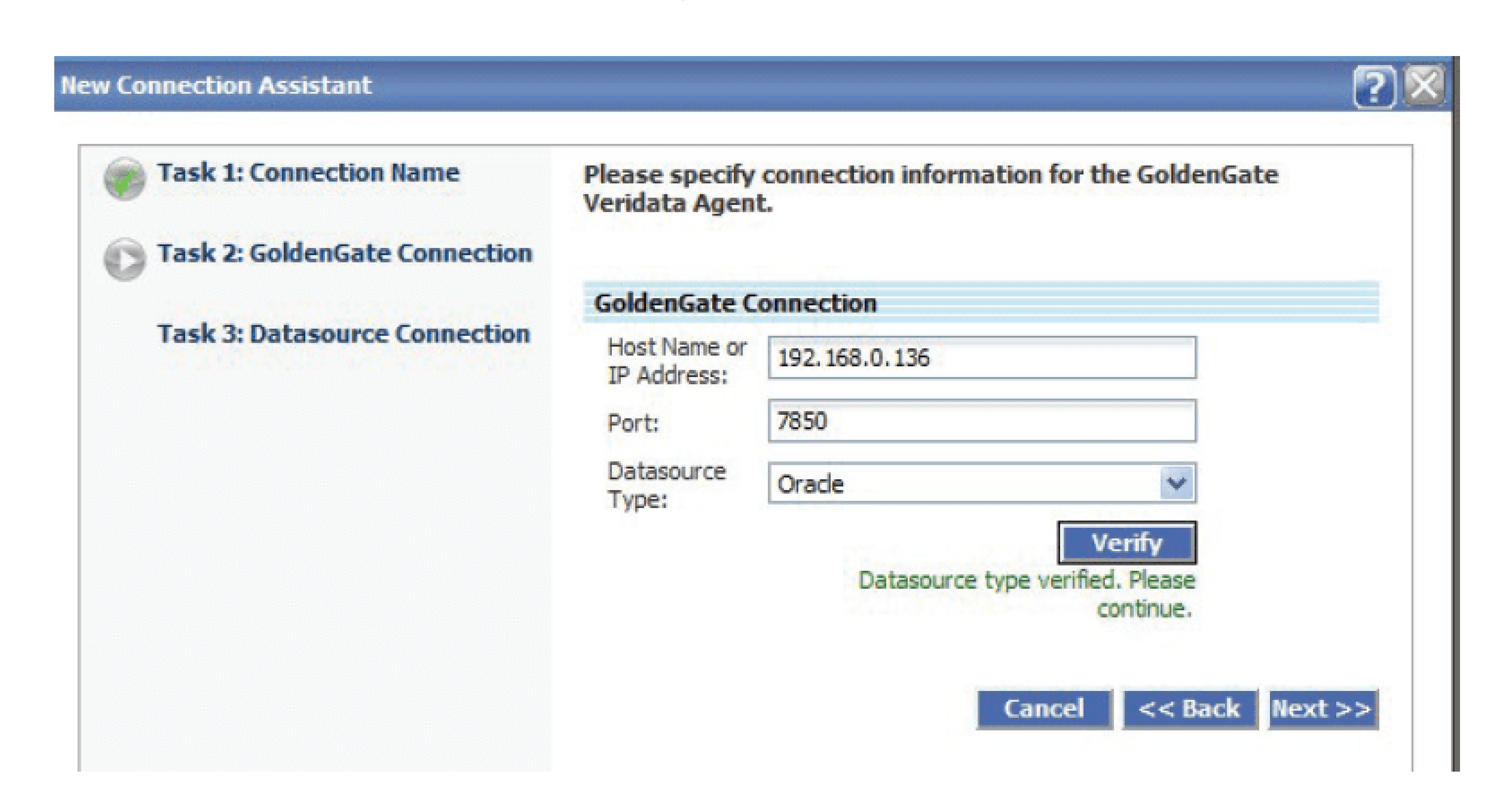


图 12-16

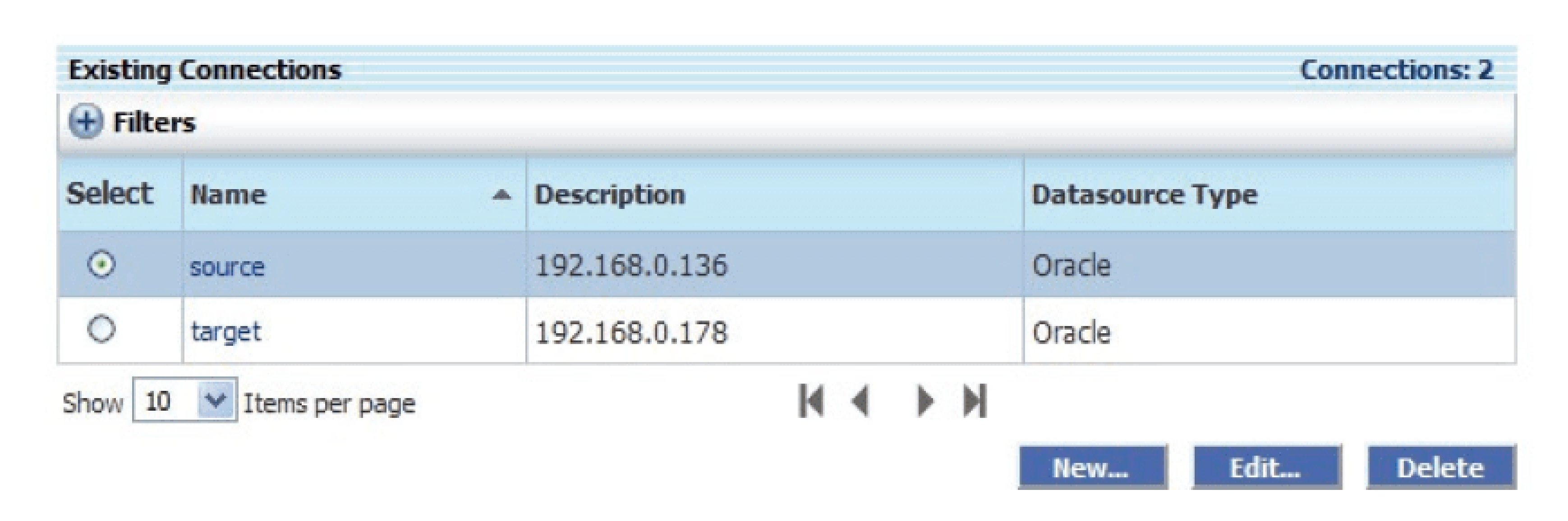


图 12-17

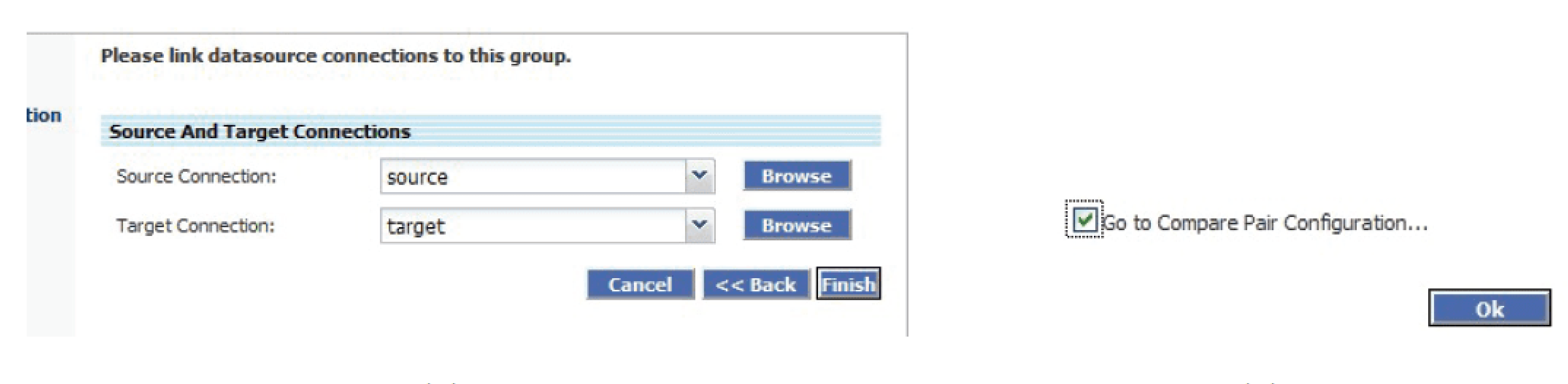


图 12-18　　图 12-19

（3）配置好组之后，紧接着需要配置 Compare Pairs，用来匹配源端与目标端比较对象的信息，如图 12-20 所示。

Compare Pair Configuration

Group Information

Group Name: group_1

Source Connection: source　Target Connection: target

Existing Compare Pairs | Pattern Mapping | Manual Mapping | Preview

Datasource Information

Source　Schema: scott　Browse

Target　Schema: scott　Browse

Pattern Compare Pair Mapping

Table Mapping Method

Map Source and Target Tables Using Exact Names

Map Source and Target Tables Using SQL % Wildcard Pattern. (Use only one % on each side.) Preview

Where Source Names Like:

Where Target Names Like:

图 12-20

（4）配置 profiles。profile 参数包括运行时间的参数以及一些比较的信息。Veridata 有一个默认的 profile，一般用它就好。

（5）配置 jobs。

当以上信息都配置完整后，就可以配置好 job 了（运行完 job，从 report 文件即可以得到需要的比较数据的结果）。

（6）最后得到相应的 report 文件。

可以利用该文件得出的结果来分析 GoldenGate 复制软件同步数据的情况，如图 12-21 所示。

Report Detail

Job Name: job1

Group Name:

Compare Pair Name:

Report Name: job1.rpt

Download Job Reports

Download This Report

```
************************************************************************
                  Oracle GoldenGate Veridata Server
                           Version 3.0.0.2
                              Build 001

Copyright (C) 2004, 2009, Oracle and/or its affiliates. All rights reserved.

                   Starting at 2011-02-21 13:05:02

************************************************************************

Microsoft Windows 2000  Workstation
Version 5.1 (Build 2600: Service Pack 3)
Process id: 2252, Thread id : 2932


Profile: new

General:
************************************************************************
     Out-Of-Sync Output Format: binary
```

图 12-21

第 13 章　GoldenGate 性能调整与优化

对于软件，性能调整与优化一直是一门很深的学问，也是最难的一部分，对于 GoldenGate，当然也不例外。

GoldenGate 的性能瓶颈主要体现在复制进程 Replicat 的入库速度。因为在容灾端网数据库里写数据时，是在执行逻辑的 SQL 语句，所以非常消耗资源。

总的来说影响 GoldenGate 性能的因素包括主机 CPU、内存、磁盘的 I/O、网络以及目标端 DB 的性能。

就细节来说，GoldenGate 的 Extract 性能依赖于主机 CPU、内存、redo log 上的磁盘 I/O、数据库的业务特点以及表拆分的粒度等。

GoldenGate 的 Pump 进程的性能与优化主要体现在网络的带宽、系统对 GoldenGate Pump 的传输限制，以及网络客观的影响（噪声、丢包率等）。

GoldenGate 的 Replicat 进程主要来自于 Replicat 进程的投递速度和业务的特点，以及容灾库的性能和表拆分的粒度等。

GoldenGate 的性能作用在系统的最上层，下面每一层的短板都会对 GoldenGate 性能造成影响。做这样一件极富成就感的事情，也是非常享受的一个过程，下面就开始性能优化之旅吧。

13.1　目标概述

本章节从源端到目标端，依次讲解每部分进程的性能调整与优化。按 GoldenGate 传输数据的顺序，首先介绍 Extract 进程的调优，然后介绍 Pump 进程的调优，最后介绍 Replicat 进程的调优。根据前面的描述，主要把重心放在 Replicat 的调优上。

13.2　Extract 进程优化

对于 Extract 进程，影响性能的因素包括主机 CPU、内存、redo log 上的磁盘 I/O、数据库的业务特点以及表拆分的粒度等。而要对 Extract 进程进行性能的调整，首先需要找到 Extract 进程的瓶颈在哪里。

查找性能瓶颈的时候需要通过操作系统级的工具，例如，人们熟悉的任务管理器、top、topas、sar、vmstat、glance iostat、vmstat 等；其次，在 GoldenGate 级别则可以通过：

示例 13-1：

```
GGSCI > stats <进程名>
```

示例 13-2：

```
GGSCI > view report <进程名>
```

来找到性能的相关信息。

而 Extract 的瓶颈一般在于 LCR 转换为 UDF 的环节上。找到性能瓶颈的环节后，则需要进行性能调整。通常的做法就是对其进程再进行拆分，对抽取进程加一些优化的参数。

注：生产环境中，抽取进程一般很少会出现性能瓶颈问题，事实上 GoldenGate 的 Extract 进程日常处理能力已经很高效，足以应付绝大多数交易量非常庞大的数据库。

13.2.1 拆分 Extract 进程

对于 Extract 进程，一般需要了解其应用的特点，根据应用的特点进行粒度的拆分。一般的做法是同一个 schema 下的表尽量放到一个进程组里，如果该 schema 的业务还是很大的话，就尽量把业务相同或者相近的表放到同一个进程里。

抽取进程的参数文件里的 table 可以用 SQL 语句来匹配出来，这也是比较高效的做法：

示例 13-3：

```
sql> select 'TABLE  '||OWNER||'.'||TABLE NAME||';' from dba tables where
owner in ('指定的 schema 名');
```

将该 select 语句结果放到抽取进程的参数文件里即可。

13.2.2 Extract 进程调优参数

对于 Extract 进程，对 GoldenGate 调优主要是对 GoldenGate Extract 进程参数的调优，根据生产系统的情况需要加入不同的参数，或者参数的值都要做相应的改变，例如：EOFDELAY、FLUSHSECS。

13.2.3 I/O 瓶颈优化

当用系统或者 GoldenGate 级别的工具来监控 GoldenGate 时，如果发现 GoldenGate 的瓶颈在 I/O，则需要对 GoldenGate 的 I/O 进行优化，包括以下几个方面。

1. 增大日志读取间隔

增大日志读取间隔：

示例 13-4：

```
EOFDELAY 3 //间隔 3s，缺省为 1s
```

控制 Extract 每 3s 读取一次日志，在 RAC 防止节点时间同步，给其留 3s 的时间。

2. 增大内存刷新间隔

增大内存刷新间隔：

示例 13-5：

```
FLUSHSECS 3 //间隔为 3s，缺省为 1s
```

例如：某生产库抽取进程常用的参数如图 13-1 所示。

```
EXTRACT ext_epma
SETENV (NLS_LANG="AMERICAN_AMERICA.ZHS16GBK")
USERID goldengate, PASSWORD *****
GETTRUNCATES
REPORTCOUNT EVERY 30 MINUTES, RATE
NUMFILES 5000
DISCARDFILE ./dirrpt/ext_epma.dsc, APPEND, MEGABYTES 300
DISCARDROLLOVER  AT 3:00
WARNLONGTRANS 2h, CHECKINTERVAL 3m
EXTTRAIL ./dirdat/ga, MEGABYTES 200
DYNAMICRESOLUTION
TRANLOGOPTIONS  EXCLUDEUSER goldengate
TRANLOGOPTIONS  CONVERTUCS2CLOBS
TRANLOGOPTIONS  ALTARCHIVELOGDEST PRIMARY instance pmhapdb1 /oracle/epm/
THREADOPTIONS   MAXCOMMITPROPAGATIONDELAY 60000
THREADOPTIONS   IOLATENCY 1000
```

图 13-1

13.3 Pump 进程组的优化

Pump 进程组负责将源端的队列文件通过网络投递到目标端主机。虽然很少有 Pump 进程出现性能瓶颈的情况（网络延迟除外），但是针对 Pump 进程也有不少可供挖掘的调优参数。

如果 GoldenGate 性能瓶颈出现在 GoldenGate 的 Pump 进程相关的这一环节，则需要对 Pump 进程组进行优化。

13.3.1 网络带宽较低优化

对一些网络带宽比较低的用户，可以通过在数据传输的过程中进程数据压缩，或者在传输前对数据进行过滤，让 Pump 进程只传输有用的数据，减轻网络的负载。这些方式都是通过减少 Pump 进程的传输量来实现调优的。

另外强调一点就是每个用于复制的表最好都有主键或者唯一索引，其生成的日志量就会少很多，这样也能减小很大一部分网络负载。

另外，还可以在操作系统级别对网络参数进行一些调整，以提高传输的效率，具体可向操作系统厂商咨询。

1. 使用数据压缩特性

如网络带宽较低，GoldenGate 提供对传输数据的压缩功能，可以使传输的数据缩小几倍，大大减少了网络带宽的需求量。

只需要在 Pump 进程中加入 compress 参数就可以实现在传输工程中对数据的压缩。当然压缩和解压会对主机的 CPU 的性能有一定影响，建议在网络带宽足够的情况下不要使用压缩特性。图 13-2 则可以说明。

例如：

```
EXTRACT dpeepma
RMTHOST 101.108.192.18, MGRPORT 7809, COMPRESS
PASSTHRU
NUMFILES 5000
RMTTRAIL ./dirdat/ra
DYNAMICRESOLUTION
```

图 13-2

2. 增大 TCP 缓存

增大 TCP 缓存大小：

示例 13-6：

```
RMTHOST test, MGRPORT 7809, TCPBUFSIZE 100000, TCPFLUSHBYTES 300000
```

注：此参数主要是把 TCP 原有的报文缓存起来，稍后发送；此参数目前不建议在生产环境下使用。

13.3.2 I/O 瓶颈

当 Pump 进程组出现 I/O 瓶颈时，应根据需要调整相应的调优参数值。

1. 增大队列读取间隔

增大队列读取间隔，减小 I/O 操作的次数来减小 I/O 的压力：

示例 13-7：

```
EOFDELAY 3 //间隔 3s，缺省为 1s
```

2. 增大内存刷新间隔

增大内存刷新间隔：

示例 13-8：

```
FLUSHSECS 5 //间隔为 5s，缺省为 1s
```

注：如果可能，Pump 进程应尽量使用调优参数 PASSTHRU 避免与源数据库交互，可以节省大量源端的主机资源。图 13-3 则可以说明。

例如：

```
EXTRACT dpeepma
RMTHOST 201.108.192.18, MGRPORT 7809, COMPRESS
PASSTHRU
NUMFILES 5000
RMTTRAIL ./dirdat/ra
DYNAMICRESOLUTION

TABLE   EPM_HA.*;
```

图 13-3

13.3.3　数据过滤与转换优化

当数据 GoldenGate 有数据过滤与数据转换时，都会增加其主机的开销。

对于数据过滤与数据转换这一块，可以做的调优工作包括以下几个方面。

（1）如果是容灾环境，生产端相对来说要重要得多，可以考虑将数据过滤与转换的工作放在投递或者复制进程去做，从而保证对生产环境的影响降到最低。

（2）当使用投递或者复制进程进行数据的过滤或者转换工作时，不妨参考以下几点原则。

- 如果源库确实能承受更多的资源开销，则把过滤或者转换参数写到投递进程参数文件里，让投递进程来实现数据过滤与数据转换，这样可以节省网络带宽，也可以保证数据的安全，因为只有转换过了的数据才投递到目标库。
- 如果目标库库能承受更多的资源开销，则把过滤或者转换参数写到复制进程参数文件里，让复制进程来实现数据过滤与数据转换。
- 另外还可以对进程进行拆分，来提升单个进程的数据处理能力。

13.4　Replicat 进程组的优化

Replicat 进程的性能调整是整个 GoldenGate 里最重要的一个环节。因为 GoldenGate 复制进程出现性能瓶颈的状况会很多，涉及到的细节也很多，瓶颈大多出现在 SQL 写回数据库，从而需要调整的工作量也很多。

下面详细介绍各个环节的各个部分。

13.4.1　操作合并

了解业务的特点后，根据业务的实际情况对复制进程实现操作合并。

复制进程操作合并的语法如下：

示例 13-9：

```
BATCHSQL
[BATCHERRORMODE | NOBATCHERRORMODE]
```

```
[BATCHESPERQUEUE <n>]
[BATCHTRANSOPS <n>]
[BYTESPERQUEUE <n>]
[OPSPERBATCH <n>]
[OPSPERQUEUE <n>]
```

例如：

示例 13-10：

```
BATCHSQL BATCHESPERQUEUE 100, OPSPERBATCH 8000
```

说明：此例中参数对于列特别多或者字段特别长的表反而可能降低性能；对于少量表重复进行操作的场景，如批处理，比较有效。可以通过对两个子参数的组合进行尝试获取最佳性能。

13.4.2 小交易合并

如果数据库的业务特点是有大量的小事务，则可以使用小交易合并这个参数。使用 GROUPTRANSPOS 这个参数可以控制每一次提交事务中的 SQL 语句的数量，从而达到优化的目的。

这样做的优势主要体现在以下两个方面。

第一：在 Replicat 端减少事务的数量。

第二：减少写 checkpoint file 或 checkpoint table 的次数。

数据库也可以合并小交易，使 SESSION 处理高效率状态。

例如：

示例 13-11：

```
GROUPTRANSOPS 1000
```

如了解到数据业务有大量的 insert 操作，则可以根据需要使用 insertappend 参数加速插入速度。

13.4.3 大交易分拆

与小交易合并对应的是大交易拆分。如果数据库的业务主要是大交易，例如大量事务都在上万条记录之上，则需要在复制进程对交易进行拆分，提高复制进程的写入速度。

大交易的拆分可以将大的交易拆成小的交易，这个参数用在数据库层面没有做最大事务的限制。如果数据库的回滚段不能容纳源端的大事务，可以通过 MAXTRANSOPS 这个参数强制拆成多个事务，来保证数据的正常复制。图 13-4 则可以说明。

例如：

Grouptransops 示例：

3 个交易，分别有 25、50、60 个记录，假如 Grouptransops 为 100（缺省值），则 Replicat 一直要等待到第 3 个交易（25 + 50 + 60）>100 时才后 Commit。

```
REPLICAT repepma
USERID goldengate, PASSWORD  *****
SETENV (NLS_LANG = "AMERICAN_AMERICA.ZHS16GBK")
REPORT AT 00:59
REPORTCOUNT EVERY 30 MINUTES, RATE
REPORTROLLOVER AT 02:00
REPERROR DEFAULT, ABEND
NUMFILES 5000
GROUPTRANSOPS 10000
MAXTRANSOPS 10000
ASSUMETARGETDEFS
DISCARDFILE ./dirrpt/repepma.dsc, APPEND, MEGABYTES 5200
DISCARDROLLOVER AT 02:00
GETTRUNCATES
ALLOWNOOPUPDATES
CHECKSEQUENCEVALUE
MAP EPM_HA.*, TARGET  EPM_HA.*;
```

图 13-4

两参数关系算法如下：

示例 13-12：

```
if end of transaction OR num of operations >= maxtransops then
if num of operations >= grouptransops then
COMMIT transaction
```

例如：

假设 Grouptransops = 100, maxtransops = 1，则上述例子中同样要等待第 3 个交易后提交。

假设 Grouptransops = 1, maxtransops = 1，则每个交易提交一次。

假设 Grouptransops = 10, maxtransops = 10，则每 10 个记录提交一次。

建议二者设置为相同值 MAXTRANSOPS = GROUPTRANSOPS。

13.4.4　拆分 Replicat 进程

当用监控工具发现 Replicat 存在性能瓶颈时，如果 Replicat 进程的拆分粒度又不够细，最好的调优方法则是对 Replicat 进程进行拆分。

1．拆分粒度

可以通过 schema 进行区分，每个复制链路负责一个或多个 schema；也可以根据表进行分割，每个进程负责不同表的集合；对于同一个表也可以通过 Range 拆分为几个进程同时处理。

例如：

示例 13-13：

```
Replicat 1:
MAP sales.acct, TARGET sales.acct, FILTER (@RANGE (1, 2));
```

```
Replicat 2:
MAP sales.acct, TARGET sales.acct, FILTER (@RANGE (2, 2));
```

2. 进程拆分的注意事项

- 各进程间没有同步机制，应尽量确保同一交易涉及表在同一个进程。
- 单个 Extract 进程可处理日志一般为 30 ~ 50GB/h，单个 Replicat 进程一般只能处理 1GB 队列/h，可采用一个 Extract 对多个 Replicat 的模式。
- 由于 Extract 在 catch up（追赶）模式需要读取归档日志，速度慢且耗费资源高，建议 Extract 一旦出现较大延迟则立即进行拆分。

3. 保证抽取一致性

由于 GoldenGate 的 Extract 性能较高，可以使用尽量少的 Extract 完成抽取，多个之间以业务或 Schema 进行区分。

单个 Extract 抽取出来的队列中可以保证交易的一致性和先后顺序。

4. 尽量保证投递一致性

如单个 Replicat 无法满足一个队列投递数据要求，可以根据表进行分割，每个进程负责不同表的集合，尽量保证同一业务涉及表放在一个 Replicat 中，可以保证一致性（需临时禁止表间的外键链接保证 Replicat 可进行拆分）。

对于同一个表也可以通过 Range 拆分为几个进程同时处理。

Replicat 拆分可能临时造成各进程间不同步，但是，多个 Replicat 性能会得到很大提高，可以保证数据复制始终是实时的。

当源端出现灾难后，由于 Extract 可以保证源端抽取时数据的一致性，而目标端多 Replicat 读取的是同一个队列，当它们应用队列数据完毕后是可以达到数据一致的。

由于 GoldenGate 是按顺序生成队列的。每个队列都有一个序列号，而且队列中的记录都是由 RBA 号唯一指定，这就使得 GoldenGate 在 Replicat 进程出现瓶颈的时候，可以很好地去拆分 Replicat 进程。

拆分 GoldenGate Replicat 在生产环境中用到的概率非常高。因为 GG 的抽取进程每小时大约可以抽取 30～50GB 的日志量，而 Replicat 进程每小时大约只可以投递 1GB 的日志量。所以在业务高峰期的时候，数据库会产生大量的日志。如果在 GoldenGate 的抽取极限内，GoldenGate 可以完全的抽取这些日志量。

而在 GoldenGate 的 Pump 进程把这些日志传送到容灾端以后，由于容灾端的机器配置一般不如生产端的高。再加上与 Replicat 相比 Extract 本身的入库速度就比较慢。这会导致日志在容灾端大量的堆积，Replicat 出现延时。这就需要对 GoldenGate 的 Replicat 进程进行拆分，来增加入库的速度。

拆分也有一定的限度，不可以无止境的去拆分。因为每个 Replicat 都会占用系统的内存资源，而且会占用 CPU 资源。所以如果拆分太多的 Replicat，把系统的内存消耗完了以后，反而会导致入库的速度更慢。在拆分的时候要根据实际情况进行有效的拆分。

下面介绍一个 Replicat 进程拆分的案例：

（1）STEP 1 停止要拆分的进程。

示例 13-14：

```
GGSCI (OE5) 7> stop repma

Sending STOP request to REPLICAT REPMA…
Request processed.
```

（2）STEP 2 查看要拆分进程的 SEQNO 和 RBA 号，

示例 13-15：

```
GGSCI (OE5) 9> info repma

REPLICAT   REPMA            Last Started 2011-03-27 10*46   Status STOPPED
Checkpoint Lag              00*00*00 (updated 00*01*58 ago)
Log Read Checkpoint         File ./dirdat/ma000001
                            2011-03-27 10*48*24.264723  RBA 117954
```

说明当前的进程读到了第 1 个队列，而且 RBA 号为 117954 的位置。

（3）STEP 3 为要新建的进程编辑参数文件，这里叫做 repmb。

示例 13-16：

```
GGSCI (OE5) 14> edit params repmb

REPLICAT repmb
USERID GoldenGate, PASSWORD GoldenGate
setenv (NLS LANG="AMERICAN AMERICA.ZHS16GBK")
--REPORT AT 01*59
REPORTCOUNT EVERY 30 MINUTES, RATE
REPERROR DEFAULT, abend
numfiles 50000
DBOPTIONS ALLOWUNUSEDCOLUMN
MAXTRANSOPS   500000
GROUPTRANSOPS 10000
CHECKPOINTSECS 40
--HANDLECOLLISIONS
assumetargetdefs
DISCARDFILE ./dirrpt/repmb.dsc, APPEND, MEGABYTES 1000
GETTRUNCATES
ALLOWNOOPUPDATES

map scott.dept , target scott.dept ;
```

（4）STEP 4 添加 repmb 进程组，并为其添加 trail 文件。

示例 13-17：

```
GGSCI (OE5) 16> ADD REPLICAT repmb ,EXTTRAIL ./dirdat/mb checkpointtable
GoldenGate.ckpt
REPLICAT added.
```

（5）STEP 5 修改 repmb 的指针和 repma 相同。
示例 13-18：

```
GGSCI (OE5) 19> alter repmb , extseqno 1 extrba 117954
REPLICAT altered.
```

（6）STEP 6 再次确认 repma 和 repmb 的指针已经修改到一致状态。
示例 13-19：

```
GGSCI (OE5) 21> info repma

REPLICAT   REPMA        Last Started 2011-03-27 10*46   Status STOPPED
Checkpoint Lag          00*00*00 (updated 00*13*25 ago)
Log Read Checkpoint     File ./dirdat/ma000001
                        2011-03-27 10*48*24.264723  RBA 117954
GGSCI (OE5) 22> info    repmb
REPLICAT   REPMB        Initialized   2011-03-27 11*04   Status STOPPED
Checkpoint Lag          00*00*00 (updated 00*01*32 ago)
Log Read Checkpoint     File ./dirdat/mb000001
                        First Record  RBA 117954
```

（7）STEP 7 一定要记住要把 repma 中拆分到 repmb 中的表排除掉，否则会有重复数据。
示例 13-20：

```
GGSCI (OE5) 23> edit params repma
tableexclude scott.dept
```

（8）STEP 8 启动 repma 和 repmb。
示例 13-21：

```
GGSCI (OE5) 23> start er *
```

到这里 GoldenGate 就把比较慢的进程查分到了不同的两个进程里面。入库就会明显地加快。

对 GoldenGate 调优其实就是对一些参数做相应的调整。入库比较慢就要对 Replicat 进程进行拆分，而且很多时候对 GoldenGate 的瓶颈就是 Replicat 进程，大多数时候调优发生在容灾端。

第 4 篇

资　料　篇

第 14 章　GoldenGate 实施的相关准备工作

本章是对 GoldenGate 实施开始之前需要做的一些准备工作的模板，主要是包括对操作系统、数据库、GoldenGate、应用等各方面信息的收集，确定哪些 GoldenGate 能做，哪些不能做。

这一步骤虽然比较简单，但是对整个实施起着非常重要的作用，甚至直接关系到实施的顺利和成败。良好的开端是成功的一半，所以请务必重视前期的准备。

14.1　前期准备的注意事项

前期准备可以避免很多不必要的错误，因此在实施之前可以尽量完成以下准备工作。

14.1.1　操作系统环境变量

HP-UX：

示例 14-1：

```
export LD_LIBRARY_PATH=/ggs/11.1*$LD_LIBRARY_PATH
```

AIX：

示例 14-2：

```
export LIBPATH=/ggs/11.1*$LIBPATH
```

这里要注意，为保险起见，可以将 GoldenGate 安装目录放在 LD_LIBRARY_PATH/LIBPATH 环境变量最前面。

14.1.2　GoldenGate 运行操作系统用户

建议使用 Oracle 用户运行 GoldenGate，Oracle 用户已将 Oracle 相关环境变量设置好，包括 lib 相关的环境变量。如果使用独立的 GoldenGate 用户，可以将 Oracle 用户的环境变量复制到 profile 文件中，额外添加 lib 包相关的内容。

14.1.3　操作系统资源使用限制

使用操作系统命令：

示例 14-3：

```
ulimit -a
```

查看相应的系统资源使用限制，通过修改/etc/security/limits 文件，建议都设置 unlimitied。

如果不能全部设置为 unlimited，建议将以下参数设置为 unlimited：

示例 14-4：

```
max memory size         (kbytes, -m)  unlimited
file size              (blocks, -f)  unlimited
data seg size           (kbytes, -d)  unlimited
cpu time               (seconds, -t) unlimited
stack size             (kbytes, -s)  unlimited
```

很多情况下出现问题，和 stack 没有设置成 unlimited 有关。

14.1.4　源数据库必须启动归档模式并开启附加日志

1．Oracle 数据库

Oracle 数据库开启归档模式，数据库需要重启，步骤如下：

示例 14-5：

```
SQL> SELECT LOG MODE FROM V$DATABASE;
SQL> SHUTDOWN IMMEDIATE
SQL> STARTUP MOUNT;
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST='/U01/archive';
SQL> ALTER DATABASE ARCHIVELOG;
SQL> ALTER DATABASE OPEN;
SQL> ARCHIVE LOG LIST;
SQL> ALTER SYSTEM SWITCH LOGFILE;
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;
SQL> SELECT SUPPLEMENTAL_LOG_DATA_MIN FROM V$DATABASE;
```

2．DB2 数据库

无论 DB2 8 还是 DB2 9 系列都需要重启数据库。

DB2 8 系列：必须设置 USEREXIT 和 LOGRETAIN 参数。

示例 14-6：

```
db2 update db cfg for <database name> using USEREXIT ON
db2 update db cfg for <database name> using LOGRETAIN ON
```

DB2 9 系列：可以不使用 USEREXIT 和 LOGRETAIN 参数，而是使用 LOGARCHMETH1 和 LOGARCHMETH2 参数开启归档模式，但是 LOGRETAIN 仍需设置为 ON。

14.1.5 C++ 运行环境的版本

1. AIX XL C 版本

AIX 5.3 需要 XL C/C++ Runtime v10.1 以上以及 libpthreads version 5.3.0.51 或以上，libpthreads 属于基础包，基本都会满足。

AIX 5.2 需要 XL C/C++ Runtime v9.0 以上以及 libpthreads version 5.2.0.106 或以上，libpthreads 属于基础包，基本都会满足。

很多情况 XL C 版本为 9.0 以下，此时必须升级 XL C 版本。

2. Windows C++版本

这里要注意，GoldenGate 的版本是和数据库和操作系统相关的，如果是 64 位的 OS，结果安装了 32 位的数据库，必须使用 32 位的 GoldenGate 和 32 位的 C++ 2005 sp1 redistributable package。

32 位下载地址：

http*//www.microsoft.com/downloadS/details.aspx?displaylang=zh-cn&FamilyID=200b2fd9-ae1a-4a14-984d-389c36f85647

64 位下载地址：

http*//www.microsoft.com/downloads/en/details.aspx?displaylang=en&FamilyID=eb4ebe2d-33c0-4a47-9dd4-b9a6d7bd44da

14.1.6 GoldenGate 安装目录

OGG 安装目录建议在存储阵列上，但是在 mount 时需注意，不能使用并发参数，例如：在 AIX 下，不能使用 CIO 参数。如果使用已有的 mount 点，并且使用并发参数，必须新建文件系统，重新 mount，作为 GoldenGate 安装目录。

14.1.7 RAC 相关设置

必须保证 GoldenGate 可以访问到 RAC 环境任何一个节点的归档日志，可以采用并行文件系统或 NFS，一般采用 NFS。

在 Extract 参数文件中指定参数：

示例 14-7：

```
TRANLOGOPTIONS altarchivelogdest primary instance ins1 /archive01,
altarchivelogdest  instance  ins2 /archive02
```

14.1.8　压缩传输设置

在 Pump 参数文件中指定压缩参数，否则将占用较大的网络带宽，导致传输时间过长。

示例 14-8：

```
RMTHOST 10.100.xx.xx, MGRPORT 7839, compress
```

14.1.9　待复制表名设置

在参数文件中建议使用具体要复制的表名称，例如，schema.table_name 的方式，便于后期维护。

14.1.10　队列文件保存期限设置

在 mgr.prm 文件中进行设置：

示例 14-9：

```
PURGEOLDEXTRACTS ./dirdat/*,usecheckpoints, minkeepdays 3
```

如果空间不够，可将 minkeepdays 修改为 MINKEEPFILES MINKEEPHOURS。

如果空间仍然紧张，仍要求立即释放空间，可修改为 MINKEEPFILES，将值设置为 1，即只保留一个处理过的队列文件。

14.1.11　抽取及复制分组

针对大的应用系统，需对要复制的表进行分组，不过分组的原则最好与应用系统开发商进行沟通，确定分组的粒度并尽量按照同一组的表业务相近或相似的原则进行。

14.1.12　AIX 使用裸设备

在 Extract 参数文件中必须添加：

示例 14-10：

```
TRANLOGOPTIONS  RAWDEVICEOFFSET  0
```

14.1.13　同步表清单

如果需要做全库的同步，不需要同步一些系统级用户的内容，只需要同步应用对应的用户下的表。下列用户为系统本身使用的用户，例如：

示例 14-11：

```
'SYS', 'SYSTEM', 'DBSNMP','SYSMAN','OUTLN','MDSYS','ORDSYS','EXFSYS',
'DMSYS','WMSYS', 'CTXSYS','ANONYMOUS','XDB','ORDPLUGINS','OLAPSYS'
```

14.1.14 临时表排除

很多人喜欢在参数文件中使用 shcema.*的方式，这样有很多临时表也会进行复制，实际上很多临时表根本不需要复制，因此如果可能，尽量排除一些不需要的临时表。

14.1.15 进程中表的拆分

表的拆分主要针对大型的数据库，通常是 TB 级别数据量，并且存在大量的大事务则需拆分，通常，Extract 不需要进行拆分，data Pump 主要是网络问题不需要改动，Replicat 进程如果事务大，事务执行慢是需要拆分的。

表拆分的一般原则如下。

table 的拆分方法一般有两种：根据业务或者事务量。

根据业务的拆分，通常是所相同的业务放在一个进程中复制，Oracle 数据库通常是一个用户就是一个业务，而其他的数据库大多数都是一个库一个业务，这种拆分可以让不同之间相互复制而毫无影响。

根据日志量的拆分，通常大型的数据库有部分表频繁地更新，导致整体性能下降，这时就要考虑这部分频繁更新的对象放在单独的队列中，这样不会对整体产生影响，并且能让这部分更新频繁的对象，更快地复制。

14.2 生产库的信息收集

14.2.1 确认要收集的信息

总结相关信息见表 14-1。

表 14-1

科目	说明
操作系统的版本	操作系统版本、CPU、内存、硬盘空间以及应用
C++运行环境的版本	C++ 运行环境是否满足要求
数据库的版本	数据库的版本和 GoldenGate 平台版本要匹配
数据库的备份策略	GoldenGate 要求数据库必须在归档模式
数据库的归档位置	GoldenGate 要求可以看到所有节点的归档信息
数据库的不支持的对象	GoldenGate 对特殊对象是否支持
数据库的复制范围确认	需要向客户确认，不能确认的话，就全部复制

14.2.2　生成信息收集的 SQL 语句

- 确认复制范围用户表的数量。
- 确认复制范围用户数据量。
- 确认复制范围有没有不支持的列。
- 确认复制表的列长度没有超过限制。
- 确认复制范围中压缩表、cluster 表。
- 确认每天产生的日志量对各种要复制的数据库对象的理解。

14.2.3　DML 与 DDL 操作

通俗地理解 DML 的操作就是改变表的数据，DDL 的操作就是改变表、触发器，存储过程等的结构。

1. GoldenGate 对数据类型的支持

相关信息见表 14-2。

表 14-2

数据类型	详细
numeric types	NUMBER、BINARY FLOAT、BINARY DOUBLE
Character types	CHAR、VARCHAR2、LONG、NCHAR、NVARCHAR2
Multi-byte character types	NCHAR、NVARCHAR2
Binary data types	RAW、LONG RAW
Date and timestamp data types	DATE、TIMESTAMP
Large object data types	CLOB、NCLOB、BLOB
Other supported data types	ROWID、VARRAY、INTERVAL DAY、INTERVAL YEAR
Non-supported data types	ANYDATA 、 ANYDATASET 、 ANYTYPE 、 BFILE 、 UROWID BINARY_INTEGER 、 MLSLABEL 、 PLS_INTEGER 、 TIMEZONE_ABBR、TIMEZONE_REGION、URITYPE

2. GoldenGate 对 DML 的支持

相关信息见表 14-3。

表 14-3

操作	详细
Insert、update、delete	regular \index-organized \clustered tablesmaterialized views
Associated transaction control operations	regular \index-organized \clustered tablesmaterialized views

3. GoldenGate 对 DDL 的支持

GoldenGate 支持所有的 DDL 但是定义语句小于 2MB，包括 clusters、functions、indexes、Packages、procedure、tables、tablespaces、roles、sequences、synonyms、triggers、Types、views、materialized views、users。

14.3 RMAN 初始化方案

RMAN 初始化是一个非常好的选择，可以自由地选择恢复到特定的 SCN，而且保证数据一致性，也可以继续追加归档，节省 GoldenGate 追日志的时间。

14.3.1 初始化 SCN 的选择

在 GoldenGate 初始化的时候，要保证所有的事务必须在 RMAN 中或者 GoldenGate 中。通常情况下，GoldenGate 提前开启抽取，确定所有的事务都是在 GoldenGate 抽取之内，就可以选取当时的 SCN 开始备份，也可以选取当前的 SCN 开始恢复。

14.3.2 多实例库恢复到单实例库的注意事项

- 第一：参数文件的重新编写。
- 第二：控制文件的数据文件改变目录。
- 第三：存储介质的改变。

14.4 自动化脚本

1. 用户表中的数据量

可以通过数据字典的相关表字段和属性来评估表的数据量，数据的准确性依赖于统计信息的准确。相关信息如图 14-1 所示。

```
SQL> select NUM_ROWS,BLOCKS from dba_tables where OWNER in ('SCOTT');

  NUM_ROWS     BLOCKS
---------- ----------
         4          5
        14          5
         0          0
         5          5
```

图 14-1

通过类似的命令来分析表的大小及数据量，来评估每个表的数据量。

```
SQL>  select owner,segment_name,bytes,blocks,buffer_pool
      from dba_segments where owner in ('SCOTT');

OWNER       SEGMENT_NAME              BYTES     BLOCKS BUFFER_
----------- -------------------- ---------- ---------- -------
SCOTT       PK_DEPT                   65536          8 DEFAULT
SCOTT       DEPT                      65536          8 DEFAULT
SCOTT       EMP                       65536          8 DEFAULT
SCOTT       PK_EMP                    65536          8 DEFAULT
SCOTT       BONUS                     65536          8 DEFAULT
SCOTT       SALGRADE                  65536          8 DEFAULT
```

图 14-2

2．用户表中的特殊数据类型

表中是否有大字段等相关字段，由于占用空间大，数据 Extract 慢很容易影响性能，如果是图片等没有可有可无的数据，建议不复制。相关信息如图 14-3 所示。

```
SQL> select table_name ,column_name ,data_type,DATA_LENGTH
     from dba_tab_columns where owner in ('SCOTT');

TABLE_NAME       COLUMN_NAME      DATA_TYPE     DATA_LENGTH
---------------- ---------------- ------------- -----------
DEPT             DEPTNO           NUMBER                 22
DEPT             DNAME            VARCHAR2               14
DEPT             LOC              VARCHAR2               13
EMP              EMPNO            NUMBER                 22
EMP              ENAME            VARCHAR2               10
EMP              JOB              VARCHAR2                9
EMP              MGR              NUMBER                 22
EMP              HIREDATE         DATE                    7
EMP              SAL              NUMBER                 22
EMP              COMM             NUMBER                 22
EMP              DEPTNO           NUMBER                 22
```

图 14-3

容易影响性能的字段通常都是长度很大，或者是存放文档，语言信息相关的字段，主要有 CLOB、BLOB、NCLOB、LONG、LONG、RAW。

3．用户表中无主键或唯一约束的列

示例 14-12：

```
SELECT owner, table name
  FROM all tables
  WHERE owner  in ('SCOTT')
MINUS
(SELECT user1.name, obj1.name
  FROM SYS.user$ user1,
       SYS.user$ user2,
       SYS.cdef$ cdef,
       SYS.con$ con1,
       SYS.con$ con2,
       SYS.obj$ obj1,
       SYS.obj$ obj2
```

```
 WHERE user1.name  in ('SCOTT')
   AND cdef.type# = 2
   AND con2.owner# = user2.user#(+)
   AND cdef.robj# = obj2.obj#(+)
   AND cdef.rcon# = con2.con#(+)
   AND obj1.owner# = user1.user#
   AND cdef.con# = con1.con#
   AND cdef.obj# = obj1.obj#
UNION
SELECT distinct(owner), idx.table name
  FROM all_indexes idx
 WHERE idx.owner  in ('SCOTT')
   AND idx.uniqueness = 'UNIQUE')
/
```

相关信息如图 14-4 所示。

```
OWNER       TABLE_NAME
----------  ------------------------------
SCOTT       BONUS
SCOTT       SALGRADE
```

图 14-4

4．用户表中有没有不支持的数据类型

示例 14-13：

```
SELECT OWNER, TABLE NAME, COLUMN NAME, DATA TYPE
FROM all_tab_columns
WHERE OWNER  in ('SCOTT')
AND (data type in ('BFILE', 'TIMEZONE REGION', 'BINARY INTEGER',
'PLS INTEGER', 'UROWID', 'URITYPE',
 'MLSLABEL', 'TIMEZONE ABBR', 'ANYDATA', 'ANYDATASET')
or data type like 'INTERVAL%')
/
```

5．当前系统的生成日志情况

示例 14-14：

```
col "Avg Log Size" format 999,999,999
select sum (BLOCKS) * max(BLOCK_SIZE)/ count(*)"Avg Log Size" From
gV$ARCHIVED_LOG;

Prompt Table* Frequency of Log Switches by hour and day
SELECT SUBSTR(TO CHAR(FIRST TIME, 'MM-DD-YY HH:MI:SS'),1,5) DAY,
   TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),
```

```
10,2),'00',1,0)),'99') "00",
TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),
10,2),'01',1,0)),'99') "01",
TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),
10,2),'02',1,0)),'99') "02",
TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),10,
2),'03',1,0)),'99') "03",
TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),10,
2),'04',1,0)),'99') "04",
TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),10,
2),'05',1,0)),'99') "05",
TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),10,
2),'06',1,0)),'99') "06",
TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),
10,2),'07',1,0)),'99') "07",
TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),
10,2),'08',1,0)),'99') "08",
TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),
10,2),'09',1,0)),'99') "09",
TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),
10,2),'10',1,0)),'99') "10",
TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),
10,2),'11',1,0)),'99') "11",
TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),
10,2),'12',1,0)),'99') "12",
TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),
10,2),'13',1,0)),'99') "13",
TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),
10,2),'14',1,0)),'99') "14",
TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),
10,2),'15',1,0)),'99') "15",
TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),
10,2),'16',1,0)),'99') "16",
TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),
10,2),'17',1,0)),'99') "17",
TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),
10,2),'18',1,0)),'99') "18",
TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),
10,2),'19',1,0)),'99') "19",
TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),
10,2),'20',1,0)),'99') "20",
TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),
10,2),'21',1,0)),'99') "21",
TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),
10,2),'22',1,0)),'99') "22",
```

```
    TO CHAR(SUM(DECODE(SUBSTR(TO CHAR(FIRST TIME, 'MM-DD-YY HH:MI:SS'),
    10,2),'23',1,0)),'99') "23"
FROM     V$LOG HISTORY
GROUP  BY SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),1,5) ;
```

6. 信息收集的自动化脚本

示例 14-15：

```
-- This script spools the output to a file named schemaCheckOracle.out
-- Example of running the script*
-- sqlplus <userid>/<pw> @full-DB CheckOracle.sql

set null "NULL VALUE"
set feedback off
set heading off
set linesize 132
set pagesize 9999
set echo off
set verify off
set trimspool on

col table name for a30
col column_name for a30
col data_type for a15
col object type for a20
col constraint type desc for a30


spool AllSchemaCheckOracle.out

SELECT '------ System Info* '
FROM dual;
set heading on
select to_char(sysdate, 'MM-DD-YYYY HH24*MI*SS') "DateTime* " from dual
/
select banner from v$version
/

select name, log_mode "LogMode",
supplemental_log_data_min "SupLog* Min", supplemental_log_data_pk "PK",
supplemental log data ui "UI", force logging "Forced",
supplemental log data fk "FK", supplemental log data all "All",
to_char(created, 'MM-DD-YYYY HH24*MI*SS') "Created"
from v$database
/
```

```
select
platform name
from v$database
/
set heading off
SELECT '------ Objects stored in Tablespaces with Compression are not
supported in the current release of OGG '
FROM dual;
select
    TABLESPACE_NAME,
    DEF_TAB_COMPRESSION,
    ENCRYPTED,
    COMPRESS FOR
from DBA TABLESPACES
where
DEF TAB COMPRESSION <> 'DISABLED';

set heading off
SELECT '------ Distinct Object Types and their Count By Schema* '
FROM dual;
SELECT owner, object_type, count(*) total
FROM all objects
WHERE OWNER not in ('SYS', 'SYSTEM', 'DBSNMP','SYSMAN','OUTLN','MDSYS',
'ORDSYS','EXFSYS','DMSYS','WMSYS','CTXSYS','ANONYMOUS','XDB','ORDPLUGINS',
'OLAPSYS','PUBLIC')
GROUP BY object_type, owner
/

SELECT '------ Distinct Column Data Types and their Count in the Schema* '
FROM dual;
SELECT data_type, count(*) total
FROM all tab columns
WHERE OWNER not in ('SYS', 'SYSTEM', 'DBSNMP','SYSMAN','OUTLN','MDSYS',
'ORDSYS','EXFSYS','DMSYS','WMSYS','CTXSYS','ANONYMOUS','XDB','ORDPLUGINS',
'OLAPSYS','PUBLIC')
GROUP BY data type
/

SELECT '------ Tables that will Fail Add Trandata (Only an issue for Oracle
versions below Oracle 10G) in the Database '
FROM dual;
SELECT distinct(table name)
FROM dba tab columns
WHERE owner not in ('SYS', 'SYSTEM', 'DBSNMP','SYSMAN','OUTLN','MDSYS',
```

```
'ORDSYS','EXFSYS','DMSYS','WMSYS','CTXSYS','ANONYMOUS','XDB','ORDPLUGINS',
'OLAPSYS','PUBLIC')
AND column id > 32
AND table_name in
(SELECT distinct(table_name)
  FROM all tables
 WHERE owner not in ('SYS', 'SYSTEM', 'DBSNMP','SYSMAN','OUTLN','MDSYS',
'ORDSYS','EXFSYS','DMSYS','WMSYS','CTXSYS','ANONYMOUS','XDB','ORDPLUGINS',
'OLAPSYS','PUBLIC')
MINUS
(SELECT obj1.name
  FROM SYS.user$ user1,
       SYS.user$ user2,
       SYS.cdef$ cdef,
       SYS.con$ con1,
       SYS.con$ con2,
       SYS.obj$ obj1,
       SYS.obj$ obj2
 WHERE user1.name not in ('SYS', 'SYSTEM', 'DBSNMP','SYSMAN','OUTLN',
'MDSYS','ORDSYS','EXFSYS','DMSYS','WMSYS','CTXSYS','ANONYMOUS','XDB',
'ORDPLUGINS','OLAPSYS','PUBLIC')
   AND cdef.type# = 2
   AND con2.owner# = user2.user#(+)
   AND cdef.robj# = obj2.obj#(+)
   AND cdef.rcon# = con2.con#(+)
   AND obj1.owner# = user1.user#
   AND cdef.con# = con1.con#
   AND cdef.obj# = obj1.obj#
UNION
SELECT idx.table name
  FROM all indexes idx
 WHERE idx.owner not in ('SYS', 'SYSTEM', 'DBSNMP','SYSMAN','OUTLN',
'MDSYS','ORDSYS','EXFSYS','DMSYS','WMSYS','CTXSYS','ANONYMOUS','XDB',
'ORDPLUGINS','OLAPSYS','PUBLIC')
   AND idx.uniqueness = 'UNIQUE'))
/

SELECT '------ Tables With No Primary Key or Unique Index by Schema* '
FROM dual;
SELECT owner, table name
  FROM all_tables
 WHERE owner not in ('SYS', 'SYSTEM', 'DBSNMP','SYSMAN','OUTLN','MDSYS',
'ORDSYS','EXFSYS','DMSYS','WMSYS','CTXSYS','ANONYMOUS','XDB','ORDPLUGINS',
'OLAPSYS','PUBLIC')
MINUS
```

```
(SELECT user1.name, obj1.name
  FROM SYS.user$ user1,
       SYS.user$ user2,
       SYS.cdef$ cdef,
       SYS.con$ con1,
       SYS.con$ con2,
       SYS.obj$ obj1,
       SYS.obj$ obj2
 WHERE user1.name not in ('SYS', 'SYSTEM', 'DBSNMP','SYSMAN','OUTLN',
'MDSYS','ORDSYS','EXFSYS','DMSYS','WMSYS','CTXSYS','ANONYMOUS','XDB',
'ORDPLUGINS','OLAPSYS','PUBLIC')
   AND cdef.type# = 2
   AND con2.owner# = user2.user#(+)
   AND cdef.robj# = obj2.obj#(+)
   AND cdef.rcon# = con2.con#(+)
   AND obj1.owner# = user1.user#
   AND cdef.con# = con1.con#
   AND cdef.obj# = obj1.obj#
UNION
SELECT distinct(owner), idx.table_name
  FROM all indexes idx
 WHERE idx.owner not in ('SYS', 'SYSTEM', 'DBSNMP','SYSMAN','OUTLN',
'MDSYS','ORDSYS','EXFSYS','DMSYS','WMSYS','CTXSYS','ANONYMOUS','XDB',
'ORDPLUGINS','OLAPSYS','PUBLIC')
   AND idx.uniqueness = 'UNIQUE')
/

SELECT '------ Tables Defined with Rowsize > 2M in all Schemas '
FROM dual;
SELECT table name, sum(data length) row length over 2M
FROM all tab columns
WHERE owner not in ('SYS', 'SYSTEM', 'DBSNMP','SYSMAN','OUTLN','MDSYS',
'ORDSYS','EXFSYS','DMSYS','WMSYS','CTXSYS','ANONYMOUS','XDB','ORDPLUGINS',
'OLAPSYS','PUBLIC')
GROUP BY table_name
HAVING sum(data length) > 2000000
/

SELECT '------ Tables With No Primary Key or Unique Index and Column lenght
> 1M '
FROM dual;
SELECT owner, table_name
  FROM all tab columns
 WHERE owner not in ('SYS', 'SYSTEM', 'DBSNMP','SYSMAN','OUTLN','MDSYS',
'ORDSYS','EXFSYS','DMSYS','WMSYS','CTXSYS','ANONYMOUS','XDB','ORDPLUGINS',
```

```
'OLAPSYS','PUBLIC')
 group by owner, table name
 HAVING sum(data length) > 1000000
MINUS
(SELECT user1.name, obj1.name
  FROM SYS.user$ user1,
       SYS.user$ user2,
       SYS.cdef$ cdef,
       SYS.con$ con1,
       SYS.con$ con2,
       SYS.obj$ obj1,
       SYS.obj$ obj2
 WHERE user1.name not in ('SYS', 'SYSTEM', 'DBSNMP','SYSMAN','OUTLN',
'MDSYS','ORDSYS','EXFSYS','DMSYS','WMSYS','CTXSYS','ANONYMOUS','XDB',
'ORDPLUGINS','OLAPSYS','PUBLIC')
   AND cdef.type# = 2
   AND con2.owner# = user2.user#(+)
   AND cdef.robj# = obj2.obj#(+)
   AND cdef.rcon# = con2.con#(+)
   AND obj1.owner# = user1.user#
   AND cdef.con# = con1.con#
   AND cdef.obj# = obj1.obj#
UNION
SELECT idx.owner, idx.table name
  FROM all indexes idx
 WHERE idx.owner not in ('SYS', 'SYSTEM', 'DBSNMP','SYSMAN','OUTLN',
'MDSYS','ORDSYS','EXFSYS','DMSYS','WMSYS','CTXSYS','ANONYMOUS','XDB',
'ORDPLUGINS','OLAPSYS','PUBLIC')
   AND idx.uniqueness = 'UNIQUE')
/

SELECT '------ Tables With CLOB, BLOB, LONG, NCLOB or LONG RAW Columns in
ALL Schemas '
FROM dual;
SELECT OWNER, TABLE_NAME, COLUMN_NAME, DATA_TYPE
FROM all tab columns
WHERE OWNER not in ('SYS', 'SYSTEM', 'DBSNMP','SYSMAN','OUTLN','MDSYS',
'ORDSYS','EXFSYS','DMSYS','WMSYS','CTXSYS','ANONYMOUS','XDB','ORDPLUGINS',
'OLAPSYS','PUBLIC')
AND data type in ('CLOB', 'BLOB', 'LONG', 'LONG RAW', 'NCLOB')
/

SELECT '------ Tables With Columns of UNSUPPORTED Datatypes in ALL Schemas'
FROM dual;
SELECT OWNER, TABLE_NAME, COLUMN_NAME, DATA_TYPE
```

```
FROM all tab columns
WHERE OWNER not in ('SYS', 'SYSTEM', 'DBSNMP','SYSMAN','OUTLN','MDSYS',
'ORDSYS','EXFSYS','DMSYS','WMSYS','CTXSYS','ANONYMOUS','XDB','ORDPLUGINS',
'OLAPSYS','PUBLIC')
AND (data_type in ('ORDDICOM', 'BFILE', 'TIMEZONE_REGION', 'BINARY_INTEGER',
'PLS INTEGER', 'UROWID', 'URITYPE', 'MLSLABEL', 'TIMEZONE ABBR', 'ANYDATA',
'ANYDATASET', 'ANYTYPE')
or data type like 'INTERVAL%')
/

SELECT '------ Cluster, or Object Tables - ALL UNSUPPORTED - in ALL Schemas'
FROM dual;
SELECT OWNER, TABLE NAME, CLUSTER NAME, TABLE TYPE
FROM all all tables
WHERE OWNER not in ('SYS', 'SYSTEM', 'DBSNMP','SYSMAN','OUTLN','MDSYS',
'ORDSYS','EXFSYS','DMSYS','WMSYS','CTXSYS','ANONYMOUS','XDB','ORDPLUGINS',
'OLAPSYS','PUBLIC')
AND (cluster name is NOT NULL or TABLE TYPE is NOT NULL)
/

Select 'All tables that have compression enabled (which we do not support)*'
from dual;
select owner, table name
from DBA TABLES
where COMPRESSION = 'ENABLED'
/

SELECT TABLE OWNER, TABLE NAME, COMPRESSION, COMPRESS FOR
FROM ALL TAB PARTITIONS
WHERE (COMPRESSION = 'ENABLED')
/

SELECT '------ IOT (Fully support for Oracle 10GR2 (with or without overflows)
using GGS 10.4 and higher) - in All Schemas* '
FROM dual;
SELECT OWNER, TABLE NAME, IOT TYPE, TABLE TYPE
FROM all all tables
WHERE OWNER not in ('SYS', 'SYSTEM', 'DBSNMP','SYSMAN','OUTLN','MDSYS',
'ORDSYS','EXFSYS','DMSYS','WMSYS','CTXSYS','ANONYMOUS','XDB','ORDPLUGINS',
'OLAPSYS','PUBLIC')
AND (IOT_TYPE is not null or TABLE_TYPE is NOT NULL)
/

SELECT '------ Tables with Domain or Context Indexes'
FROM dual;
```

```
SELECT OWNER, TABLE NAME, index name, index type
FROM dba indexes
WHERE OWNER not in ('SYS', 'SYSTEM', 'DBSNMP','SYSMAN','OUTLN','MDSYS',
'ORDSYS','EXFSYS','DMSYS','WMSYS','CTXSYS','ANONYMOUS','XDB','ORDPLUGINS',
'OLAPSYS','PUBLIC')
and index type = 'DOMAIN'
/

SELECT '------ Types of Constraints on the Tables in ALL Schemas '
FROM dual;
SELECT DECODE(constraint_type,'P','PRIMARY KEY','U','UNIQUE', 'C',
'CHECK', 'R', 'REFERENTIAL') constraint_type_desc, count(*) total
FROM all constraints
WHERE OWNER not in ('SYS', 'SYSTEM', 'DBSNMP','SYSMAN','OUTLN','MDSYS',
'ORDSYS','EXFSYS','DMSYS','WMSYS','CTXSYS','ANONYMOUS','XDB','ORDPLUGINS',
'OLAPSYS','PUBLIC')
GROUP BY constraint type
/

SELECT '------ Cascading Deletes on the Tables in ALL Schemas '
FROM dual;
SELECT table name, constraint name
FROM all constraints
WHERE OWNER not in ('SYS', 'SYSTEM', 'DBSNMP','SYSMAN','OUTLN','MDSYS',
'ORDSYS','EXFSYS','DMSYS','WMSYS','CTXSYS','ANONYMOUS','XDB','ORDPLUGINS',
'OLAPSYS','PUBLIC')
    and constraint_type = 'R' and delete_rule = 'CASCADE'
/

SELECT '------ Tables Defined with Triggers in ALL Schema* '
FROM dual;
SELECT table_name, COUNT(*) trigger_count
FROM all triggers
WHERE OWNER not in ('SYS', 'SYSTEM', 'DBSNMP','SYSMAN','OUTLN','MDSYS',
'ORDSYS','EXFSYS','DMSYS','WMSYS','CTXSYS','ANONYMOUS','XDB','ORDPLUGINS',
'OLAPSYS','PUBLIC')
GROUP BY table name
/

SELECT '------ Performance issues - Reverse Key Indexes Defined in ALL Schema*'
FROM dual;
col Owner format a10
col TABLE OWNER format a10
col INDEX TYPE format a12
SET Heading on
```

```
select
    OWNER,
    INDEX_NAME,
    INDEX_TYPE,
    TABLE OWNER,
    TABLE NAME,
    TABLE TYPE,
    UNIQUENESS,
    CLUSTERING FACTOR,
    NUM_ROWS,
    LAST_ANALYZED,
    BUFFER POOL
from dba indexes
where index type = 'NORMAL/REV'
     And OWNER not in ('SYS', 'SYSTEM', 'DBSNMP','SYSMAN','OUTLN','MDSYS',
     'ORDSYS','EXFSYS','DMSYS','WMSYS','CTXSYS','ANONYMOUS','XDB',
     'ORDPLUGINS','OLAPSYS','PUBLIC')
/

SET Heading off
SELECT '------ Sequence numbers - Sequences could be a issue for HA
configurations '
FROM dual;

COLUMN SEQUENCE_OWNER FORMAT a15
COLUMN SEQUENCE_NAME FORMAT a30
COLUMN INCR FORMAT 999
COLUMN CYCLE FORMAT A5
COLUMN ORDER FORMAT A5
SET Heading on
SELECT SEQUENCE OWNER,
     SEQUENCE NAME,
     MIN_VALUE,
     MAX_VALUE,
     INCREMENT BY INCR,
     CYCLE FLAG CYCLE,
     ORDER FLAG "ORDER",
     CACHE SIZE,
     LAST NUMBER
  FROM DBA_SEQUENCES
 WHERE SEQUENCE_OWNER not in ('SYS', 'SYSTEM', 'DBSNMP','SYSMAN','OUTLN',
'MDSYS','ORDSYS','EXFSYS','DMSYS','WMSYS','CTXSYS','ANONYMOUS','XDB',
'ORDPLUGINS','OLAPSYS','PUBLIC')
 /
```

```
set linesize 132

col "Avg Log Size" format 999,999,999
select sum (BLOCKS) * max(BLOCK_SIZE)/ count(*)"Avg Log Size" From
gV$ARCHIVED_LOG;

Prompt Table* Frequency of Log Switches by hour and day
SELECT SUBSTR(TO CHAR(FIRST TIME, 'MM-DD-YY HH:MI:SS'),1,5) DAY,
   TO CHAR(SUM(DECODE(SUBSTR(TO CHAR(FIRST TIME, 'MM-DD-YY HH:MI:SS'),
   10,2),'00',1,0)),'99') "00",
   TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),
   10,2),'01',1,0)),'99') "01",
   TO CHAR(SUM(DECODE(SUBSTR(TO CHAR(FIRST TIME, 'MM-DD-YY HH:MI:SS'),
   10,2),'02',1,0)),'99') "02",
   TO CHAR(SUM(DECODE(SUBSTR(TO CHAR(FIRST TIME, 'MM-DD-YY HH:MI:SS'),10,
   2),'03',1,0)),'99') "03",
   TO CHAR(SUM(DECODE(SUBSTR(TO CHAR(FIRST TIME, 'MM-DD-YY HH:MI:SS'),10,
   2),'04',1,0)),'99') "04",
   TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),10,
   2),'05',1,0)),'99') "05",
   TO CHAR(SUM(DECODE(SUBSTR(TO CHAR(FIRST TIME, 'MM-DD-YY HH:MI:SS'),
   10,2),'06',1,0)),'99') "06",
   TO CHAR(SUM(DECODE(SUBSTR(TO CHAR(FIRST TIME, 'MM-DD-YY HH:MI:SS'),
   10,2),'07',1,0)),'99') "07",
   TO CHAR(SUM(DECODE(SUBSTR(TO CHAR(FIRST TIME, 'MM-DD-YY HH:MI:SS'),
   10,2),'08',1,0)),'99') "08",
   TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),10,
   2),'09',1,0)),'99') "09",
   TO CHAR(SUM(DECODE(SUBSTR(TO CHAR(FIRST TIME, 'MM-DD-YY HH:MI:SS'),
   10,2),'10',1,0)),'99') "10",
   TO CHAR(SUM(DECODE(SUBSTR(TO CHAR(FIRST TIME, 'MM-DD-YY HH:MI:SS'),
   10,2),'11',1,0)),'99') "11",
   TO CHAR(SUM(DECODE(SUBSTR(TO CHAR(FIRST TIME, 'MM-DD-YY HH:MI:SS'),
   10,2),'12',1,0)),'99') "12",
   TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),
   10,2),'13',1,0)),'99') "13",
   TO CHAR(SUM(DECODE(SUBSTR(TO CHAR(FIRST TIME, 'MM-DD-YY HH:MI:SS'),
   10,2),'14',1,0)),'99') "14",
   TO CHAR(SUM(DECODE(SUBSTR(TO CHAR(FIRST TIME, 'MM-DD-YY HH:MI:SS'),
   10,2),'15',1,0)),'99') "15",
   TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),
   10,2),'16',1,0)),'99') "16",
   TO CHAR(SUM(DECODE(SUBSTR(TO CHAR(FIRST TIME, 'MM-DD-YY HH:MI:SS'),
   10,2),'17',1,0)),'99') "17",
   TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),
```

```
    10,2),'18',1,0)),'99') "18",
    TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),
    10,2),'19',1,0)),'99') "19",
    TO CHAR(SUM(DECODE(SUBSTR(TO CHAR(FIRST TIME, 'MM-DD-YY HH:MI:SS'),
    10,2),'20',1,0)),'99') "20",
    TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),
    10,2),'21',1,0)),'99') "21",
    TO CHAR(SUM(DECODE(SUBSTR(TO CHAR(FIRST TIME, 'MM-DD-YY HH:MI:SS'),
    10,2),'22',1,0)),'99') "22",
    TO_CHAR(SUM(DECODE(SUBSTR(TO_CHAR(FIRST_TIME, 'MM-DD-YY HH:MI:SS'),
    10,2),'23',1,0)),'99') "23"
FROM     V$LOG HISTORY
GROUP  BY SUBSTR(TO CHAR(FIRST TIME, 'MM-DD-YY HH:MI:SS'),1,5) ;

SELECT '------ Summary of log volume processed by day for last 7 days* '
FROM dual;
select to char(first time, 'mm/dd') ArchiveDate,
      sum(BLOCKS*BLOCK_SIZE/1024/1024) LOGMB
from v$archived_log
where first time > sysdate - 7
group by to char(first time, 'mm/dd')
order by to_char(first_time, 'mm/dd');
/

SELECT '------ Summary of log volume processed per hour for last 7 days* '
FROM dual;
select to_char(first_time, 'MM-DD-YYYY') ArchiveDate,
      to char(first time, 'HH24') ArchiveHour,
      sum(BLOCKS*BLOCK_SIZE/1024/1024) LogMB
from v$archived log
where first_time > sysdate - 7
group by to char(first time, 'MM-DD-YYYY'), to char(first time, 'HH24')
order by to char(first time, 'MM-DD-YYYY'), to char(first time, 'HH24');
/

set heading off
select '* This output may be found in file* AllSchemsCheckOracle.out' from
dual
/

spool off
undefine b0
-- exit
```

第 15 章　GoldenGate 认证操作系统及数据库矩阵

经 GoldenGate 当前最新版本 11.1 官方认证的数据库及操作系统版本比较多，现将归纳的内容列入表 15-1。

表 15-1

数据库	版本	位	操作系统	版本	硬件平台
Enscribe	N/A	N/A	HP Nonstop	D46-D48&G06	NSK
Enscribe	N/A	N/A	HP Nonstop	H06	NSK
IBM DB2 UDB	8.1	32	AIX	5.2	IBM PowerPC
IBM DB2 UDB	8.1	64	AIX	5.2	IBM PowerPC
IBM DB2 UDB	8.1	32	AIX	5.3	IBM PowerPC
IBM DB2 UDB	8.1	64	AIX	5.3	IBM PowerPC
IBM DB2 UDB	8.1	32	AIX	5.1	IBM PowerPC
IBM DB2 UDB	8.1	64	AIX	5.1	IBM PowerPC
IBM DB2 UDB	8.1	64	RedHat AS	4.0	AMD/Intel x64
IBM DB2 UDB	8.1	64	RedHat AS	3.0	AMD/Intel x64
IBM DB2 UDB	8.1	32	RedHat AS	3.0	Intel x86
IBM DB2 UDB	8.1	32	RedHat AS	4.0	Intel x86
IBM DB2 UDB	8.1	32	Solaris	8	Sun SPARC
IBM DB2 UDB	8.1	64	Solaris	8	Sun SPARC
IBM DB2 UDB	8.1	64	Solaris	9	Sun SPARC
IBM DB2 UDB	8.1	64	Solaris	10	Sun SPARC
IBM DB2 UDB	8.1	N/A	z/OS	1.04	Mainframe
IBM DB2 UDB	8.1	N/A	z/OS	1.06	Mainframe
IBM DB2 UDB	8.2	32	AIX	5.1	IBM PowerPC
IBM DB2 UDB	8.2	64	AIX	5.1	IBM PowerPC
IBM DB2 UDB	8.2	32	AIX	5.2	IBM PowerPC
IBM DB2 UDB	8.2	64	AIX	5.2	IBM PowerPC
IBM DB2 UDB	8.2	32	AIX	5.3	IBM PowerPC
IBM DB2 UDB	8.2	64	AIX	5.3	IBM PowerPC
IBM DB2 UDB	8.2	64	RedHat AS	3.0	AMD/Intel x64
IBM DB2 UDB	8.2	32	RedHat AS	3.0	Intel x86

续表

数据库	版本	位	操作系统	版本	硬件平台
IBM DB2 UDB	8.2	64	RedHat AS	4.0	AMD/Intel x64
IBM DB2 UDB	8.2	32	RedHat AS	4.0	Intel x86
IBM DB2 UDB	8.2	64	Solaris	9	Sun SPARC
IBM DB2 UDB	8.2	32	Solaris	8	Sun SPARC
IBM DB2 UDB	8.2	64	Solaris	8	Sun SPARC
IBM DB2 UDB	8.2	64	Solaris	10	Sun SPARC
IBM DB2 UDB	8.2	32	SuSE	9	Intel x86
IBM DB2 UDB	9.1 / 9.5	64	AIX	5.3	IBM PowerPC
IBM DB2 UDB	9.1 / 9.5	64	AIX	6.1	IBM PowerPC
IBM DB2 UDB	9.1 / 9.5	64	Solaris	10	Sun SPARC
IBM DB2 UDB	9.1 / 9.5	64	Windows	2000	AMD/Intel x64
IBM DB2 UDB	9.1 / 9.5	64	Windows	2003	AMD/Intel x64
IBM DB2 UDB	9.1 / 9.5	64	Windows	XP	AMD/Intel x64
MS SQL Server	2000	32	Windows	2000	Intel x86
MS SQL Server	2000	32	Windows	2003	Intel x86
MS SQL Server	2000	32	Windows	XP	Intel x86
MS SQL Server	2000	32	Windows	2003	AMD/Intel x64
MS SQL Server	2005	32	Windows	2000	Intel x86
MS SQL Server	2005	32	Windows	2003	Intel x86
MS SQL Server	2005	32	Windows	XP	Intel x86
MS SQL Server	2005	64	Windows	2003	AMD/Intel x64
MS SQL Server	2005	64	Windows	2003	Intel IA64
MS SQL Server	2008	32	Windows	XP	Intel x86
MS SQL Server	2008	32	Windows	2000	Intel x86
MS SQL Server	2008	32	Windows	2003	Intel x86
MS SQL Server	2008	64	Windows	2003	AMD/Intel x64
MS SQL Server	2008	64	Windows	2003	Intel IA64
MySQL	4.1	32	RedHat AS	2.1	Intel x86
MySQL	4.1	64	RedHat AS	2.1	Intel IA64
MySQL	4.1	64	RedHat AS	3.0	AMD/Intel x64
MySQL	4.1	32	RedHat AS	3.0	Intel x86
MySQL	4.1	64	RedHat AS	4.0	AMD/Intel x64
MySQL	4.1	32	RedHat AS	4.0	Intel x86
MySQL	4.1	32	Windows	2000	Intel x86
MySQL	4.1	32	Windows	2003	Intel x86
MySQL	4.1	32	Windows	XP	Intel x86
MySQL	4.1	64	Windows	2003	AMD/Intel x64
MySQL	5.0	32	RedHat AS	2.1	Intel x86
MySQL	5.0	64	RedHat AS	3.0	AMD/Intel x64
MySQL	5.0	64	RedHat AS	4.0	AMD/Intel x64

续表

数据库	版本	位	操作系统	版本	硬件平台
MySQL	5.0	32	RedHat AS	4.0	Intel x86
MySQL	5.0	64	RedHat AS	2.1	Intel IA64
MySQL	5.0	32	RedHat AS	3.0	Intel x86
MySQL	5.0	32	Windows	2000	Intel x86
MySQL	5.0	64	Windows	2003	AMD/Intel x64
MySQL	5.0	32	Windows	2003	Intel x86
MySQL	5.0	32	Windows	XP	Intel x86
Oracle	9.2	64	AIX	5.1	IBM PowerPC
Oracle	9.2	64	AIX	5.3	IBM PowerPC
Oracle	9.2	32	AIX	5.1	IBM PowerPC
Oracle	9.2	32	AIX	5.2	IBM PowerPC
Oracle	9.2	64	AIX	5.2	IBM PowerPC
Oracle	9.2	32	AIX	5.3	IBM PowerPC
Oracle	9.2	32	AIX	6.1	IBM PowerPC
Oracle	9.2	64	AIX	6.1	IBM PowerPC
Oracle	9.2	64	HP-UX	11.23	HP Intel IA64
Oracle	9.2	64	HP-UX	11.31	HP Intel IA64
Oracle	9.2	64	HP-UX	11.11	HP PA-RISC
Oracle	9.2	32	HP-UX	11.31	HP PA-RISC
Oracle	9.2	32	HP-UX	11.11	HP PA-RISC
Oracle	9.2	32	HP-UX	11.23	HP PA-RISC
Oracle	9.2	64	Oracle Linux	4.0	AMD/Intel x64
Oracle	9.2	64	Oracle Linux	5.0	AMD/Intel x64
Oracle	9.2	32	Oracle Linux	3.0	Intel x86
Oracle	9.2	32	Oracle Linux	4.0	Intel x86
Oracle	9.2	32	Oracle Linux	5.0	Intel x86
Oracle	9.2	64	Oracle Linux	3.0	Intel IA64
Oracle	9.2	64	Oracle Linux	3.0	AMD/Intel x64
Oracle	9.2	64	Oracle Linux	4.0	Intel IA64
Oracle	9.2	64	Oracle Linux	5.0	Intel IA64
Oracle	9.2	64	Solaris	8	Sun SPARC
Oracle	9.2	64	Solaris	9	Sun SPARC
Oracle	9.2	64	Solaris	10	Sun SPARC
Oracle	9.2	64	SuSE	9	Opteron
Oracle	9.2	32	SuSE	9	Intel x86
Oracle	9.2	64	Tru64	5.1	HP OSF
Oracle	9.2	64	Windows	2003	AMD/Intel x64
Oracle	9.2	32	Windows	2000	Intel x86
Oracle	9.2	32	Windows	2003	Intel x86
Oracle	9.2	32	Windows	XP	Intel x86

续表

数据库	版本	位	操作系统	版本	硬件平台
Oracle	10.1	64	AIX	5.3	IBM PowerPC
Oracle	10.1	32	AIX	5.2	IBM PowerPC
Oracle	10.1	64	AIX	5.2	IBM PowerPC
Oracle	10.1	32	AIX	6.1	IBM PowerPC
Oracle	10.1	32	AIX	5.3	IBM PowerPC
Oracle	10.1	64	AIX	6.1	IBM PowerPC
Oracle	10.1	32	HP-UX	11.31	HP PA-RISC
Oracle	10.1	64	HP-UX	11.31	HP Intel IA64
Oracle	10.1	64	HP-UX	11.23	HP Intel IA64
Oracle	10.1	64	HP-UX	11.11	HP PA-RISC
Oracle	10.1	32	HP-UX	11.11	HP PA-RISC
Oracle	10.1	32	HP-UX	11.23	HP PA-RISC
Oracle	10.1	64	HP-UX	11.31	HP PA-RISC
Oracle	10.1	64	Oracle Linux	4.0	AMD/Intel x64
Oracle	10.1	32	Oracle Linux	3.0	Intel x86
Oracle	10.1	32	Oracle Linux	4.0	Intel x86
Oracle	10.1	64	Oracle Linux	3.0	Intel IA64
Oracle	10.1	64	Oracle Linux	3.0	AMD/Intel x64
Oracle	10.1	64	Oracle Linux	4.0	Intel IA64
Oracle	10.1	64	Solaris	8	Sun SPARC
Oracle	10.1	64	Solaris	9	Sun SPARC
Oracle	10.1	64	Solaris	10	Sun SPARC
Oracle	10.1	64	SuSE	9	Opteron
Oracle	10.1	32	SuSE	9	Intel x86
Oracle	10.1	64	Tru64	5.1	HP OSF
Oracle	10.1	64	Windows	2003	AMD/Intel x64
Oracle	10.1	32	Windows	2000	Intel x86
Oracle	10.1	32	Windows	2003	Intel x86
Oracle	10.1	32	Windows	XP	Intel x86
Oracle	10.2	64	AIX	5.2	IBM PowerPC
Oracle	10.2	64	AIX	5.3	IBM PowerPC
Oracle	10.2	32	AIX	5.2	IBM PowerPC
Oracle	10.2	32	AIX	5.3	IBM PowerPC
Oracle	10.2	64	HP-UX	11.23	HP Intel IA64
Oracle	10.2	64	HP-UX	11.31	HP Intel IA64
Oracle	10.2	64	HP-UX	11.11	HP PA-RISC
Oracle	10.2	64	HP-UX	11.23	HP PA-RISC
Oracle	10.2	64	HP-UX	11.31	HP PA-RISC
Oracle	10.2	32	HP-UX	11.11	HP PA-RISC
Oracle	10.2	32	HP-UX	11.23	HP PA-RISC

续表

数据库	版本	位	操作系统	版本	硬件平台
Oracle	10.2	32	HP-UX	11.31	HP PA-RISC
Oracle	10.2	64	Oracle Linux	3.0	AMD/Intel x64
Oracle	10.2	64	Oracle Linux	4.0	AMD/Intel x64
Oracle	10.2	64	Oracle Linux	5.0	AMD/Intel x64
Oracle	10.2	32	Oracle Linux	3.0	Intel x86
Oracle	10.2	32	Oracle Linux	4.0	Intel x86
Oracle	10.2	64	Oracle Linux	3.0	Intel IA64
Oracle	10.2	64	Oracle Linux	4.0	Intel IA64
Oracle	10.2	64	Oracle Linux	5.0	Intel IA64
Oracle	10.2	64	Solaris	8	Sun SPARC
Oracle	10.2	64	Solaris	9	Sun SPARC
Oracle	10.2	64	Solaris	10	Sun SPARC
Oracle	10.2	64	SuSE	9	Opteron
Oracle	10.2	32	SuSE	9	Intel x86
Oracle	10.2	64	Tru64	5.1	HP OSF
Oracle	10.2	64	Windows	2003	AMD/Intel x64
Oracle	10.2	32	Windows	2000	Intel x86
Oracle	10.2	32	Windows	2003	Intel x86
Oracle	10.2	32	Windows	XP	Intel x86
Oracle	11.1	64	AIX	5.3	IBM PowerPC
Oracle	11.1	32	AIX	5.3	IBM PowerPC
Oracle	11.1	64	AIX	6.1	IBM PowerPC
Oracle	11.1	64	HP-UX	11.23	HP Intel IA64
Oracle	11.1	64	HP-UX	11.31	HP Intel IA64
Oracle	11.1	64	RedHat AS	4.0	AMD/Intel x64
Oracle	11.1	32	RedHat AS	4.0	Intel x86
Oracle	11.1	64	RedHat AS	5.0	AMD/Intel x64
Oracle	11.1	64	Solaris	10	Sun SPARC
Oracle	11.1	64	Windows	2003	AMD/Intel x64
TimesTen	7.0.5	64	Oracle Linux	4.0	AMD/Intel x64
Sybase ASE	12.5.4	32	AIX	5.1	IBM PowerPC
Sybase ASE	12.5.4	64	AIX	5.1	IBM PowerPC
Sybase ASE	12.5.4	32	AIX	5.2	IBM PowerPC
Sybase ASE	12.5.4	64	AIX	5.2	IBM PowerPC
Sybase ASE	12.5.4	32	AIX	5.3	IBM PowerPC
Sybase ASE	12.5.4	64	AIX	5.3	IBM PowerPC
Sybase ASE	12.5.4	64	HP-UX	11.23	HP Intel IA64
Sybase ASE	12.5.4	32	HP-UX	11.11	HP PA-RISC
Sybase ASE	12.5.4	64	HP-UX	11.11	HP PA-RISC
Sybase ASE	12.5.4	32	HP-UX	11.23	HP PA-RISC

续表

数据库	版本	位	操作系统	版本	硬件平台
Sybase ASE	12.5.4	64	HP-UX	11.23	HP PA-RISC
Sybase ASE	12.5.4	32	RedHat AS	4.0	AMD/Intel x64
Sybase ASE	12.5.4	32	RedHat AS	4.0	Intel x86
Sybase ASE	12.5.4	32	Solaris	8	Sun SPARC
Sybase ASE	12.5.4	32	Solaris	9	Sun SPARC
Sybase ASE	12.5.4	64	Solaris	8	Sun SPARC
Sybase ASE	12.5.4	64	Solaris	9	Sun SPARC
Sybase ASE	12.5.4	64	Solaris	10	Sun SPARC
Sybase ASE	12.5.4	32	Windows	2000	Intel x86
Sybase ASE	15.0	64	RedHat AS	4.0	AMD/Intel x64
Sybase ASE	12.5.4	32	Windows	2003	Intel x86
Teradata	V2R5	32	RedHat AS	3.0	Intel x86
Teradata	V2R5	32	Windows	2000	Intel x86
Teradata	V2R5	32	Windows	2003	Intel x86
Teradata	V2R5	32	Windows	XP	Intel x86
Teradata	V2R6	32	RedHat AS	3.0	Intel x86
Teradata	V2R6	32	RedHat AS	4.0	Intel x86
Teradata	V2R6	64	RedHat AS	3.0	AMD/Intel x64
Teradata	V2R6	64	RedHat AS	4.0	AMD/Intel x64
Teradata	V2R6	32	Windows	2000	Intel x86
Teradata	V2R6	64	Windows	2003	AMD/Intel x64
Teradata	V2R6	32	Windows	2003	Intel x86
Teradata	V2R6	32	Windows	XP	Intel x86
Oracle	10.2	64	SuSE	10	Opteron
SQL/MX	NA	32	Windows	2000	Intel x86
SQL/MX	NA	32	Windows	2003	Intel x86
SQL/MX	NA	32	Windows	XP	Intel x86
SQL/MX	NA	NA	HP Nonstop	G06	NA
SQL/MX	NA	NA	HP Nonstop	H06	NA
Enscribe	N/A	N/A	HP Nonstop	D46-D48&G06	NSK
Enscribe	N/A	N/A	HP Nonstop	H06 & J06	Itanium
SQL/MX	2.3	32	Windows	2000	Intel x86
SQL/MX	2.3	32	Windows	2003	Intel x86
SQL/MX	2.3	32	Windows	XP	Intel x86
SQL/MX	2.3	N/A	HP Nonstop	G06	NSK
SQL/MX	2.3	N/A	HP Nonstop	H06 & J06	Itanium
SQL/MP	N/A	N/A	HP Nonstop	G06	NSK
SQL/MP	N/A	N/A	HP Nonstop	H06 & J06	Itanium
TimesTen	7.0.5	64	RedHat AS	4	AMD/Intel x64

后　　记

作为企业级 IT 运维系列中的《GoldenGate 企业级运维实战》如果能抛砖引玉，带给大家思路和启迪，则足以欣慰。就像我们回溯历史经历和切身之事时，经常感觉自身的渺小，就如同地球之比于宇宙，小得像太平洋上的一滴水珠。

而这个世界上，最接近永远不变的，恰恰是“变化”本身。随着产品的更新，新技术的出现，以及厂商之间的商业整合，很多具体的技术细节都会随之发生变化，届时也会跟随有相应的修订，但其中涉及的基本概念和技术思路却是相通的。道为本，术为末，始能根深叶茂。

更多实战经验书籍，请参看叱咤风云同系列的《Tuxedo 企业级运维实战》和《WebLogic 企业级运维实战》等。